Krylov-subspace recycling via the POD-augmented conjugate-gradient algorithm

Kevin Carlberg[1], Virginia Forstall[2], and Ray Tuminaro[1]

[1] *Sandia National Laboratories*
[2] *University of Maryland*

Abstract

This work presents a new Krylov-subspace-recycling method for efficiently solving sequences of linear systems of equations characterized by a non-invariant symmetric-positive-definite matrix. As opposed to typical truncation strategies used in recycling such as deflation, we propose a truncation method inspired by goal-oriented proper orthogonal decomposition (POD) from model reduction. This idea is based on the observation that model reduction aims to compute a low-dimensional subspace that contains an *accurate* solution; as such, we expect the proposed method to generate a low-dimensional subspace that is well suited for computing *inexact* solutions. In particular, we propose specific goal-oriented POD 'ingredients' that align the optimality properties of POD with the objective of Krylov-subspace recycling. To compute solutions in the resulting 'augmented' POD subspace, we propose a hybrid direct/iterative three-stage iterative method that leverages (1) the optimal ordering of POD basis vectors, and (2) well-conditioned reduced matrices. Numerical experiments performed on real-world solid-mechanics problems highlight the benefits of the proposed method over standard approaches for Krylov-subspace recycling.

1 Introduction

This work considers solving a sequence of linear systems of equations characterized by a non-invariant symmetric-positive-definite matrix. Such problems arise in a variety of engineering and science applications, including structural optimization, nonlinear structural dynamics, unconstrained numerical optimization, and nonlinear electromagnetics. In particular, we consider solving these linear systems to *inexact tolerances* using the (preconditioned) conjugate-gradient method; further, we allow for a solution-dependent output to serve as a quantity of interest.

While each linear-system solve can be executed independently of previous solves, reusing data generated during these solves can lead to improved convergence; this observation has led to the emergence of *Krylov-subspace recycling* methods. These recycling methods can also be considered 'augmented Krylov subspace methods' [23] because they 'augment' the typical Krylov subspace with a subspace computed from previous data, and subsequently project the linear system onto this augmented subspace. First, researchers developed methods that employ the space spanned by *all* Krylov vectors generated during the solution of previous linear systems as the augmenting subspace. In this framework, researchers developed block Krylov [16] and successive right-hand side [22, 11, 9] methods to treat multiple right-hand sides with an invariant matrix; the successive right-hand side case occurs, for example, when restarting Krylov-subspace methods. These ideas were also extended to solve multiple systems with non-invariant matrices; approximate orthogonalization techniques [18, 21], projection methods [10, 12], and an efficient full orthogonalization method [20] were developed for this purpose.

However, retaining the accumulation of the all previous Krylov vectors can be computationally expensive and memory intensive, particularly when convergence is slow, the number of previous linear systems is large, or the preconditioner is not based on domain decomposition. This has led to the development of *truncation methods* that retain a subset of the previous Krylov vectors as a basis for the augmenting subspace. First, deflation techniques for sequences of systems with invariant [5, 25] and non-invariant [19, 17] matrices were developed. These methods employ approximated eigenvectors associated with the smallest eigenvalues of the governing matrices as the augmenting subspace $\mathcal{Y}$. As such, they are effective primarily in cases where convergence is hampered by a few small eigenvalues. An alternative approach

computes the augmenting subspace $\mathcal{Y}$ as the subspace that most accurately represents the Krylov subspace in the orthogonalization step of the generalized conjugate residual (GCR) method; this was also developed fr both the invariant [6] and non-invariant [17] cases.

These truncated Krylov-subspace-recycling techniques do not target the efficient solution of *inexact solutions*, which is the focus of this work. To this end, we propose a new proper orthogonal decomposition (POD)-augmented conjugate-gradient algorithm for (truncated) Krylov-subspace recycling. This approach—which is employs goal-oriented model reduction to truncate previous Krylov vectors—is inspired by the observation that model-reduction techniques aim to generate low-dimensional approximations that preserve high levels of accuracy (i.e., satisfy inexact tolerances); it is based on preliminary work presented in Ref. [3]. The paper consists of the following new contributions:

1. Analyses that expose the close relationship between goal-oriented POD and Krylov-subspace recycling (Theorems 1–3 and Corollary 1).
2. Goal-oriented POD ingredients for truncating previous Krylov vectors, including
   - *Snapshots* comprising all previous Krylov vectors,
   - *Metrics* induced by (1) the system matrix and (2) the output quantity of interest, and
   - *Weights* arising from (1) the linear system before truncation and (2) a radial-basis-function approximation of the solution.
3. A novel 'three-stage' algorithm, which accelerates the solution over the augmenting space using a hybrid direct/iterative approach. The algorithm leverages (1) the optimal ordering of POD basis vectors and (2) well-conditioned reduced matrices. The algorithm comprises
   - *Stage 1*: Direct solution over the first few (high-energy) POD basis vectors,
   - *Stage 2*: Iterative solution over the full augmenting space using the augmented CG algorithm, and
   - *Stage 3*: Iterative solution over the full space using the augmented preconditioned CG algorithm. This stage is equipped with new strategies for efficiently orthogonalizing against the entire augmenting subspace.

Section 2 provides the problem formulation, Section 3 describes the proposed POD-augmented conjugate-gradient algorithm, Section 4 describes the three-stage algorithm, Section 5 provides numerical experiments, and Section 6 concludes the paper. In the remainder of this paper, we denote matrices by capitalized bold letters, vectors by lowercase bold letters, and scalars by lowercase letters. We denote the columns of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ by $\mathbf{a}_i \in \mathbb{R}^m$, $i \in \mathbb{N}(n)$ with $\mathbb{N}(a) := \{1, \dots, a\}$ such that $\mathbf{A} := [\mathbf{a}_1 \ \cdots \ \mathbf{a}_n]$. We denote the scalar-valued matrix elements by $a_{ij} \in \mathbb{R}$ such that $\mathbf{a}_j \equiv [a_{1j} \ \cdots \ a_{mj}]^T$, $j \in \mathbb{N}(n)$. In addition, we denote the range of a matrix by its calligraphic counterpart, i.e., $\text{range}(\mathbf{A}) \equiv \mathcal{A}$; we sometimes refer to $\mathbf{A}$ as the 'basis' for $\mathcal{A}$, although—more precisely—it is the 'basis in matrix form'.

2 Problem formulation

This work considers solving a sequence of linear systems with a non-invariant matrix

$$\mathbf{A}_j \mathbf{x}_j^\star = \mathbf{b}_j, \quad j = 1, \dots, p. \tag{1}$$

Here, $\mathbf{A}_j \in \text{SPD}(n)$ and $\mathbf{b}_j \in \mathbb{R}^n$ denote the jth sparse system matrix and right-hand side, respectively, with $\text{SPD}(n)$ denoting the set of symmetric-positive-definite (SPD) $n \times n$ matrices. The quantities $\mathbf{x}_j^\star \in \mathbb{R}^n$ are implicitly defined as the (exact) solutions to Eqs. (1). Further, we assume that the primary objective is to compute an output quantity of interest $\mathbf{z}(\mathbf{x}_j)$, $j = 1, \dots, p$ with

$$\mathbf{z} : \mathbb{R}^n \to \mathbb{R}^q \tag{2}$$

$$\mathbf{x} \mapsto \mathbf{C}\mathbf{x}, \tag{3}$$

where $\mathbf{C} \in \mathbb{R}^{q\times n}$.

We consider computing a sequence of *inexact* solutions $\mathbf{x}_j$, $j = 1, \dots, p$ to Eqs. (1) that satisfy

$$\|\mathbf{b}_j - \mathbf{A}_j \mathbf{x}_j\|_2 \leq \epsilon_j, \quad j = 1, \dots, p. \tag{4}$$

where $\epsilon_j \geq 0$, $j = 1, \dots, p$ denotes the forcing sequence [8]. To compute each inexact solution, we consider applying the preconditioned conjugate gradient (PCG) algorithm, which computes a sequence

of solutions that minimize the energy norm of the error over the (current) Krylov subspace. For the jth linear system, these solutions satisfy

$$\mathbf{x}_j^{(k)} = \arg \min_{\mathbf{x} \in \mathbf{x}_j^{(0)} + K_j^{(k)}(\mathbf{x}_j^{(0)})} \left\| \mathbf{x}_j^\star - \mathbf{x} \right\|_{\mathbf{A}_j}, \quad k = 1, \ldots, k_j, \tag{5}$$

where $K_j^{(k)} : \mathbf{x} \mapsto K^{(k)}\left(\mathbf{M}_j^{-1}\mathbf{A}_j, \mathbf{M}_j^{-1}\left(\mathbf{b}_j - \mathbf{A}_j\mathbf{x}\right)\right) \subseteq \mathbb{R}^n$ is the affine search subspace, $K^{(k)} : (\mathbf{A}, \mathbf{b}) \mapsto \operatorname{span}\{\mathbf{b}, \mathbf{A}\mathbf{b}, \ldots, \mathbf{A}^{k-1}\mathbf{b}\} \subseteq \mathbb{R}^n$ is the Krylov subspace at iteration k, $\mathbf{M}_j \in \mathrm{SPD}(n)$ is a preconditioner, $\mathbf{x}_j^{(0)} \in \mathbb{R}^n$ is the initial solution, $(\mathbf{x}, \mathbf{y})_{\mathbf{A}_j} := \mathbf{x}^T\mathbf{A}_j\mathbf{y}$ and $\|\mathbf{x}\|_{\mathbf{A}_j} := \sqrt{(\mathbf{x}, \mathbf{x})_{\mathbf{A}_j}}$ denote the $\mathbf{A}_j$-weighted inner product and norm, and k_j denotes the number of iterations required such that $\mathbf{x}_j := \mathbf{x}_j^{(k_j)}$ satisfies inequality (4).

Note that Eqs. (5) can be equivalently expressed [24] as a projection

$$\mathbf{x}_j^{(k)} = \mathbf{P}_{\mathbf{A}_j}^{K_j^{(k)}}\left(\mathbf{x}_j^\star\right), \quad k = 1, \ldots, k_j, \tag{6}$$

where $\mathbf{P}_{\mathbf{A}}^{K}(\mathbf{x})$ denotes the $\mathbf{A}$-orthogonal projection of the vector $\mathbf{x} \in \mathbb{R}^n$ onto the (affine) subspace $K \subseteq \mathbb{R}^n$, i.e., $\mathbf{P}_{\mathbf{A}}^{K}(\mathbf{x}) \in K,\ \forall \mathbf{x} \in \mathbb{R}^n$; $\left(\mathbf{x} - \mathbf{P}_{\mathbf{A}}^{K}(\mathbf{x}), \mathbf{z}\right)_{\mathbf{A}} = 0,\ \forall \mathbf{x} \in \mathbb{R}^n,\ \forall \mathbf{z} \in K$; and $\mathbf{P}_{\mathbf{A}}^{K}\left(\mathbf{P}_{\mathbf{A}}^{K}(\mathbf{x})\right) = \mathbf{P}_{\mathbf{A}}^{K}(\mathbf{x})$ (idempotency). Given a basis $\mathbf{V} \in \mathbb{R}^{n \times y}$ and centering point $\bar{\mathbf{x}} \in \mathbb{R}^n$ for the subspace such that $K := \bar{\mathbf{x}} + \mathcal{V}$ (recall that $\mathcal{V} \equiv \operatorname{range}(\mathbf{V})$), the projector can be defined algebraically as

$$\mathbf{P}_{\mathbf{A}}^{K} : \mathbf{x} \mapsto \bar{\mathbf{x}} + \mathbf{V}\left(\mathbf{V}^T\mathbf{A}\mathbf{V}\right)^{-1}\mathbf{V}^T\mathbf{A}(\mathbf{x} - \bar{\mathbf{x}}). \tag{7}$$

If a symmetric factorization $\mathbf{A} = (\mathbf{A}^{1/2})^T\mathbf{A}^{1/2}$ is available (where $\mathbf{A}^{1/2}$ need not be upper triangular), then an equivalent definition is

$$\mathbf{P}_{\mathbf{A}}^{K} : \mathbf{x} \mapsto \bar{\mathbf{x}} + \mathbf{V}\left(\mathbf{A}^{1/2}\mathbf{V}\right)^{+}\mathbf{A}^{1/2}(\mathbf{x} - \bar{\mathbf{x}}). \tag{8}$$

where a superscript $+$ denotes the Moore–Penrose pseudoinverse.

Substituting definition (7) in Eq. (6) and using Eq. (1) as well as the definition of $K_j^{(k)}$ reveals that the solution $\mathbf{x}_j^{(k)}$ can be computed from a Galerkin projection process

$$\mathbf{V}_j^{(k)T}\mathbf{A}_j\mathbf{V}_j^{(k)}\hat{\mathbf{x}}_j^{(k)} = \mathbf{V}_j^{(k)T}\mathbf{r}_j^{(0)}, \tag{9}$$

$$\mathbf{x}_j^{(k)} = \mathbf{x}_j^{(0)} + \mathbf{V}_j^{(k)}\hat{\mathbf{x}}_j^{(k)}, \quad k = 1, \ldots, k_j \tag{10}$$

where $\mathbf{V}_j^{(k)} \in \mathbb{R}_\star^{n \times k}$ constitutes a basis for the subspace $K^{(k)}\left(\mathbf{M}_j^{-1}\mathbf{A}_j, \mathbf{M}_j^{-1}\mathbf{r}_j^{(0)}\right)$ such that $\operatorname{range}(\mathbf{V}_j^{(k)}) = K^{(k)}\left(\mathbf{M}_j^{-1}\mathbf{A}_j, \mathbf{M}_j^{-1}\mathbf{r}_j^{(0)}\right)$ and $\mathbf{r}_j^{(0)} := \mathbf{b}_j - \mathbf{A}_j\mathbf{x}_j^{(0)} \in \mathbb{R}^n$ is the initial residual. Here, $\mathbb{R}_\star^{m \times n}$ denotes the noncompact Stiefel manifold: the set of full-column-rank $m \times n$ matrices.

2.1 Augmented conjugate-gradient method

Krylov-subspace recycling aims to reduce the computational burden of solving linear system j of Eqs. (1) by exploiting a previously computed 'augmenting' subspace $\mathcal{Y}_j \subseteq \mathbb{R}^n$ of dimension $y_j \leq n$ spanned by a basis $\mathbf{Y}_j \in \mathbb{R}_\star^{n \times y_j}$. To achieve this, recycling strategies employ augmented Krylov-subspace methods; rather than perform the typical sequence of Krylov iterations (e.g., Eqs. (5)), these methods first compute a solution in the augmenting subspace, and then compute an increment in a newly generated Krylov subspace. Critically, these methods ensure that the final solution minimizes the error over the sum of augmenting and Krylov subspaces; this generally requires maintaining orthogonality of new Krylov vectors (or $\mathbf{A}_j$-orthogonality of new search directions) to the augmenting subspace.

In the context of the conjugate-gradient method, these methods first solve a minimization problem in the affine subspace $\bar{\mathbf{x}}_j + \mathcal{Y}_j$, where $\bar{\mathbf{x}}_j$ is an initial guess:

$$\mathbf{x}_j^{\mathcal{Y}_j} = \arg \min_{\mathbf{x} \in \bar{\mathbf{x}}_j + \mathcal{Y}_j} \left\| \mathbf{x}_j^\star - \mathbf{x} \right\|_{\mathbf{A}_j}. \tag{11}$$

As before, the solution can be expressed as an orthogonal projection

$$\mathbf{x}_j^{\mathcal{Y}_j} = \bar{\mathbf{x}}_j + \mathbf{Y}_j\hat{\mathbf{y}}_j = \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j + \mathcal{Y}_j}\left(\mathbf{x}_j^\star\right), \tag{12}$$

where $\hat{\mathbf{y}}_j$ satisfies

$$\mathbf{Y}_j^T \mathbf{A}_j \mathbf{Y}_j \hat{\mathbf{y}}_j = \mathbf{Y}_j^T \left(\mathbf{b}_j - \mathbf{A}_j \bar{\mathbf{x}}_j \right). \tag{13}$$

Subsequently, these methods solve a minimization problem in the 'augmented' Krylov subspace

$$\mathbf{x}_j^{(k)} = \arg \min_{\mathbf{x} \in \bar{\mathbf{x}}_j + \mathcal{Y}_j + K_j^{(k)}(\mathbf{x}_j^{\mathcal{Y}_j})} \left\| \mathbf{x}_j^\star - \mathbf{x} \right\|_{\mathbf{A}_j}, \quad k = 1, \ldots, k_j. \tag{14}$$

The final solution can then be expressed as

$$\mathbf{x}_j = \bar{\mathbf{x}}_j + \mathbf{Y}_j \hat{\mathbf{y}}_j + \mathbf{V}_j \hat{\mathbf{v}}_j = \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j + \mathcal{Y}_j + \mathcal{V}_j} \left(\mathbf{x}_j^\star \right), \tag{15}$$

where $\mathbf{V}_j := \mathbf{V}_j^{(k_j)}$ denotes a basis that satisfies $\mathcal{Y}_j + \mathcal{V}_j = \mathcal{Y}_j + K^{(k)} \left(\mathbf{M}_j^{-1} \mathbf{A}_j, \mathbf{M}_j^{-1} \left(\mathbf{b}_j - \mathbf{A}_j \mathbf{x}_j^{\mathcal{Y}_j} \right) \right)$.

The solution increment $\mathbf{V}_j \hat{\mathbf{v}}_j$ in Eq. (15) can be computed via Algorithm 1 as

$$(k_j, \hat{\mathbf{v}}_j, \mathbf{V}_j, \mathbf{\Gamma}_j) = \texttt{augmented\_pcg}(\mathbf{A}_j, \mathbf{b}_j - \mathbf{A}_j \bar{\mathbf{x}}_j, \hat{\mathbf{y}}_j, \mathbf{Y}_j, \mathbf{M}_j, \epsilon_j).$$

This algorithm is a general augmented preconditioned CG algorithm, where steps 4–5, and 13–14 account for the augmenting subspace by ensuring all search directions remain $\mathbf{A}$-orthogonal to the augmenting subspace. Important properties of this algorithm include

$$\mathbf{V}^T \mathbf{A} \mathbf{V} = \mathbf{\Gamma}, \quad \mathbf{Y}^T \mathbf{A} \mathbf{V} = \mathbf{0}, \tag{16}$$

$$\mathcal{Y} + \mathcal{V} = \mathcal{Y} + K^{(k)} \left(\mathbf{M}^{-1} \mathbf{A}, \mathbf{M}^{-1} \left(\mathbf{b} - \mathbf{A} \mathbf{Y} \hat{\mathbf{x}}^{(0)} \right) \right) \tag{17}$$

Furthermore, if $\mathbf{Y} \hat{\mathbf{x}}^{(0)} = \mathbf{P}_{\mathbf{A}}^{\bar{\mathbf{x}}_j + \mathcal{Y}} (\mathbf{x}^\star)$, then

$$\mathbf{Y} \hat{\mathbf{x}}^{(0)} + \mathbf{V} \hat{\mathbf{v}} = \mathbf{P}_{\mathbf{A}}^{\mathcal{Y} + \mathcal{V}} \left(\mathbf{x}^\star \right), \tag{18}$$

where $\mathbf{A} \mathbf{x}^\star = \mathbf{b}$.

Algorithm 1 `augmented_pcg`

Input: $\mathbf{A}$, $\mathbf{b}$, $\hat{\mathbf{x}}^{(0)}$, $\mathbf{Y}$, $\mathbf{M}$, ϵ
Output: k, $\hat{\mathbf{v}}$, $\mathbf{V}$, $\mathbf{\Gamma}$

1: $\mathbf{x}^{(0)} = \mathbf{Y} \hat{\mathbf{x}}^{(0)}$
2: $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A} \mathbf{x}^{(0)}$
3: $\mathbf{z}^{(0)} = \mathbf{M}^{-1} \mathbf{r}^{(0)}$
4: Solve $\mathbf{Y}^T \mathbf{A} \mathbf{Y} \boldsymbol{\mu}^{(0)} = \mathbf{Y}^T \mathbf{A} \mathbf{z}^{(0)}$.
5: $\mathbf{p}^{(0)} = \mathbf{z}^{(0)} - \mathbf{Y} \boldsymbol{\mu}^{(0)}$
6: **for** $k = 0, 1, \ldots$ **do**
7: $\quad \gamma^{(k)} = (\mathbf{A} \mathbf{p}^{(k)}, \mathbf{p}^{(k)})$
8: $\quad \alpha^{(k)} = (\mathbf{r}^{(k)}, \mathbf{z}^{(k)}) / \gamma^{(k)}$
9: $\quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)} \mathbf{p}^{(k)}$
10: $\quad \mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha^{(k)} \mathbf{A} \mathbf{p}^{(k)}$
11: $\quad \mathbf{z}^{(k+1)} = \mathbf{M}^{-1} \mathbf{r}^{(k+1)}$
12: $\quad \beta^{(k+1)} = \frac{(\mathbf{r}^{(k+1)}, \mathbf{z}^{(k+1)})}{(\mathbf{r}^{(k)}, \mathbf{z}^{(k)})}$
13: $\quad$ Solve $\mathbf{Y}^T \mathbf{A} \mathbf{Y} \boldsymbol{\mu}^{(k+1)} = \mathbf{Y}^T \mathbf{A} \mathbf{z}^{(k+1)}$.
14: $\quad \mathbf{p}^{(k+1)} = \mathbf{z}^{(k+1)} + \beta^{(k+1)} \mathbf{p}^{(k)} - \mathbf{Y} \boldsymbol{\mu}^{(k+1)}$
15: $\quad$ **if** $\|\mathbf{r}^{(k+1)}\| \leq \epsilon$ **then**
16: $\quad\quad$ Exit.
17: $\quad$ **end if**
18: **end for**
19: $k = k + 1$, $\hat{\mathbf{v}} = \begin{bmatrix} \alpha^{(0)} & \cdots & \alpha^{(k-1)} \end{bmatrix}^T$, $\mathbf{V} = \begin{bmatrix} \mathbf{p}^{(0)} & \cdots & \mathbf{p}^{(k-1)} \end{bmatrix}$, $\mathbf{\Gamma} = \mathrm{diag}(\gamma^{(0)}, \ldots, \gamma^{(k-1)})$

Several strategies exist to obtain the augmenting matrix $\mathbf{Y}_j$. Typically, the columns of this matrix consist of all Krylov vectors generated for previous linear systems; this provides the interpretation of 'recycling' Krylov subspaces. In this case, we have

$$\mathbf{Y}_j = \begin{bmatrix} \mathbf{V}_1 & \cdots & \mathbf{V}_{j-1} \end{bmatrix}. \tag{19}$$

However, after a modest number of linear systems has been solved, it becomes memory- and computation-intensive to retain and orthogonalize against this complete set of Krylov vectors. Therefore, several techniques have been devised to truncate these vectors to preserve the subspace that is 'most important' in some sense. In particular, these methods compute $\mathbf{Y}_{j+1}$ such that $\mathcal{Y}_{j+1} \subseteq \mathcal{Z}_j$, where $\mathbf{Z}_j := [\mathbf{Y}_j \ \mathbf{V}_j] \in \mathbb{R}^{n \times z_j}$ denotes the matrix of all preserved vectors accumulated over the first j linear solves, with $z_j = y_j + k_j$. Note that $\mathbf{Z}_{j-1} = [\mathbf{V}_1 \ \cdots \ \mathbf{V}_{j-1}]$ before truncation first occurs.

2.2 Deflation with harmonic Ritz vectors

Deflation techniques aim to retain the subspace associated with eigenvectors that tend to hamper convergence, i.e., those with eigenvalues close to zero. Such techniques have been developed for multiple linear systems with an invariant (e.g., restarting) [5, 25] and non-invariant [19, 17] matrix.

To accomplish this, these techniques compute the harmonic Ritz vectors—which approximate the smallest eigenvalues of $\mathbf{A}_j$—by solving the following problem: Find $\bar{\mathbf{y}}_i \in \text{range}(\mathbf{A}_{j-1}\mathbf{Z}_{j-1})$ and $\lambda_i \in \mathbb{R}$, $i = 1, \ldots, y_j$ with $y_j \leq z_{j-1}$ such that

$$\left(\boldsymbol{w}, \mathbf{A}_{j-1}^{-1}\bar{\mathbf{y}}_i - \bar{\mathbf{y}}_i\lambda_i\right) = 0, \quad \forall \boldsymbol{w} \in \text{range}(\mathbf{A}_{j-1}\mathbf{Z}_{j-1}). \tag{20}$$

An equivalent problem statement is the following: Find $\mathbf{y}_i \in \mathcal{Z}_{j-1}$ and $\theta_i \in \mathbb{R}$, $i = 1, \ldots, y_j$ with $y_j \leq z_{j-1}$ such that

$$(\boldsymbol{w}, \mathbf{A}_{j-1}\mathbf{y}_i - \mathbf{y}_i\theta_i) = 0, \quad \forall \boldsymbol{w} \in \text{range}(\mathbf{A}_{j-1}\mathbf{Z}_{j-1}), \tag{21}$$

where $\theta_i = \lambda_i^{-1}$ and $\bar{\mathbf{y}}_i = \mathbf{A}_{j-1}\mathbf{y}_i$, $i = 1, \ldots, y_{j+1}$. Because vector $\mathbf{y}_i$ is equivalent to eigenvector $\bar{\mathbf{y}}_i$ with one step of inverse iteration, vectors $\mathbf{y}_i$ are typically employed for recycling [15].

Algebraically, this corresponds to solving the generalized eigenvalue problem

$$\mathbf{Z}_{j-1}^T\mathbf{A}_{j-1}^T\mathbf{A}_{j-1}\mathbf{Z}_{j-1}\mathbf{G} = \mathbf{Z}_{j-1}^T\mathbf{A}_j\mathbf{Z}_{j-1}\mathbf{G}\boldsymbol{\Lambda}^{-1} \tag{22}$$

with $\boldsymbol{\Lambda} = \text{diag}(\lambda_i)$ and subsequently setting $\mathbf{Y}_j \leftarrow \left[\mathbf{Z}_{j-1}\mathbf{g}_1 \ \cdots \ \mathbf{Z}_{j-1}\mathbf{g}_{y_j}\right]$, where the eigenvalues λ_i are ordered in decreasing magnitude.

This approach has been pursued in the context of GMRES with deflated restarting (GMRES-DR [5, 19, 20, 14]), GCR with orthogonalization and deflated restarting (GCRO-DR) [17], and the deflated conjugate gradient method [5]. Deflation is effective primarily in cases where the matrix is characterized by a small number of eigenvalues close to zero that hamper convergence. Further, because this approach aims to promote convergence (to the 'exact' solution), it is not tailored for the efficient computation of *inexact* solutions, which is the focus of this work. To this end, we propose a novel truncation strategy inspired by model reduction.

3 POD-augmented CG

Because this work focuses on efficiently computing *inexact* solutions, we aim to compute an 'ideal' low-dimensional subspace $\mathcal{Y}_j^\star$ that directly minimizes the error in the augmenting-subspace solution appearing in Eq. (11), i.e.,

$$\mathcal{Y}_j^\star = \arg\min_{\mathcal{Y} \in \mathcal{G}\left(y_j, n\right)} \left\| \mathbf{x}_j^\star - \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j + \mathcal{Y}_j}\left(\mathbf{x}_j^\star\right) \right\|_{\mathbf{A}_j}, \tag{23}$$

where $\mathcal{G}\left(m, n\right)$ denotes the set of m-dimensional subspaces of $\mathbb{R}^n$ (the Grassmannian). Alternatively, we may be interested in computing inexact solutions that most accurately represent output quantities of interest. In this case, we can also consider an ideal output-oriented subspace

$$\bar{\mathcal{Y}}_j^\star = \arg\min_{\mathcal{Y} \in \mathcal{G}\left(y_j, n\right)} \|\mathbf{z}\left(\mathbf{x}_j^\star\right) - \mathbf{z}\left(\mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j + \mathcal{Y}_j}\left(\mathbf{x}_j^\star\right)\right)\|_2. \tag{24}$$

Clearly, these ideal subspaces are computable if the exact solution $\mathbf{x}_j^\star$ is known; in this case, we can enforce $\mathbf{x}_j^\star - \bar{\mathbf{x}}_j \in \mathcal{Y}_j^\star$ (resp. $\mathbf{x}_j^\star - \bar{\mathbf{x}}_j \in \bar{\mathcal{Y}}_j^\star$)—which yields $\left\|\mathbf{x}_j^\star - \mathbf{x}_j^{\mathcal{Y}_j^\star}\right\|_{\mathbf{A}_j} = \|\mathbf{z}\left(\mathbf{x}_j^\star\right) - \mathbf{z}\left(\mathbf{x}_j^{\mathcal{Y}_j^\star}\right)\|_2 = 0$—for any $y_j \geq 1$ (if $y_j = 1$, then $\mathcal{Y}_j^\star = \text{span}\{\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\}$). However, the exact solution $\mathbf{x}_j^\star$ is not known before the jth linear system is solved. In this case, we aim to compute a subspace that *approximately* solves minimization problem (23). For this purpose, we employ goal-oriented POD with carefully chosen snapshots, weights, and metrics [2, 4].

3.1 POD

The POD method generates a basis that optimally represents a set of vectors (or 'snapshots') in a certain sense. We aim to select goal-oriented POD ingredients (i.e., snapshots, weights, and metric) that align the POD optimality property with the objective function in Eq. (23).

Given a matrix whose columns represent s snapshots $\mathbf{W} \in \mathbb{R}^{n\times s}$, a vector of weights $\boldsymbol{\gamma} \in \mathbb{R}^s$, and pseudometric associated with matrix $\boldsymbol{\Theta} \in \mathrm{SPSD}(n)$, where $\mathrm{SPSD}(n)$ denotes the set of $n \times n$ symmetric-positive-semidefinite matrices, the POD method computes a subspace of dimension $y \leq s$ that minimizes the sum of squared projection errors, i.e.,

$$\mathcal{U}_y^{\boldsymbol{\Theta}}(\mathbf{W}, \boldsymbol{\gamma}) = \arg\min_{\mathcal{A}\in\mathcal{G}(y,n)} \sum_{i=1}^{s} \| \left(\mathbf{I} - \mathbf{P}_{\boldsymbol{\Theta}}^{\mathcal{A}}\right)(\gamma_i \mathbf{w}_i) \|_{\boldsymbol{\Theta}}^2, \quad y = 1, \ldots, s. \tag{25}$$

Appendix A describes two standard algorithms for computing a basis for this POD subspace $\mathbf{U}_y^{\boldsymbol{\Theta}}(\mathbf{W}, \boldsymbol{\gamma}) \in \mathbb{R}^{n\times y}$ satisfying $\mathrm{range}(\mathbf{U}_y^{\boldsymbol{\Theta}}(\mathbf{W}, \boldsymbol{\gamma})) = \mathcal{U}_y^{\boldsymbol{\Theta}}(\mathbf{W}, \boldsymbol{\gamma})$. The basis is nested

$$\mathbf{U}_{i+1}^{\boldsymbol{\Theta}} = [\mathbf{U}_i^{\boldsymbol{\Theta}} \ \mathbf{u}_{i+1}^{\boldsymbol{\Theta}}], \quad i = 1, \ldots, s-1 \tag{26}$$

with $\mathbf{u}_i^{\boldsymbol{\Theta}} \in \mathbb{R}^n$, $i = 1, \ldots, s$ and $\mathbf{U}_1^{\boldsymbol{\Theta}} = \mathbf{u}_1^{\boldsymbol{\Theta}}$ and it exhibits $\boldsymbol{\Theta}$-orthogonality, i.e.,

$$\left(\mathbf{u}_i^{\boldsymbol{\Theta}}, \mathbf{u}_j^{\boldsymbol{\Theta}}\right)_{\boldsymbol{\Theta}} = \delta_{ij}, \tag{27}$$

where δ_{ij} denotes the Kronecker delta. We note that the algebraic definition (7) is not valid when considering pseudometrics, as $\mathbf{V}^T\boldsymbol{\Theta}\mathbf{V}$ may not be invertible if $\boldsymbol{\Theta}$ is semidefinite. In this case, definition (8) is appropriate, as $\boldsymbol{\Theta}^{1/2}\mathbf{V}$ always has a pseudoinverse.

Remark 1 (Optimal ordering of POD vectors)**.** *Note that Eqs.* (25)–(26) *imply that the POD basis vectors are optimally ordered, i.e., the first y POD basis vectors span the optimal y-dimensional subspace in the sense of Eq.* (25)*.*

3.2 POD and the augmented conjugate gradient method

We propose employing goal-oriented POD to define the augmenting subspace, i.e.,

$$\mathcal{Y}_j = \mathcal{U}_y^{\boldsymbol{\Theta}}(\mathbf{W}, \boldsymbol{\gamma}) \tag{28}$$

for specific choices of the snapshots $\mathbf{W}$, weights $\boldsymbol{\gamma}$, and metric $\boldsymbol{\Theta}$. The following results provide guidance toward this end. We first show that POD is related to minimizing $\left\|\mathbf{x}_j^\star - \mathbf{x}_j^{\mathcal{Y}_j}\right\|_{\mathbf{A}_j}$, which is the error minimized by ideal subspace $\mathcal{Y}_j^\star$ in problem (23).

Theorem 1. *The POD subspace* $\mathcal{U}_y^{\mathbf{A}_j}\left(\mathbf{Z}_{j-1}, \boldsymbol{\eta}_j^\star\right)$ *with*

$$\boldsymbol{\eta}_j^\star := \left(\mathbf{Z}_{j-1}^T\mathbf{A}_j\mathbf{Z}_{j-1}\right)^{-1}\mathbf{Z}_{j-1}^T\left(\mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j\right) \tag{29}$$

minimizes an upper bound for $\left\|\mathbf{x}_j^\star - \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right)\right\|_{\mathbf{A}_j}$ *over all y-dimensional subspaces of* $\mathcal{Z}_{j-1} \subseteq \mathbb{R}^n$*, where* $y \leq z_{j-1} \leq n$*.*

Proof. We begin by decomposing the centered exact solution as $\mathbf{x}_j^\star - \bar{\mathbf{x}}_j = \mathbf{x}_j^{\|} + \mathbf{x}_j^{\perp}$, where $\mathbf{x}_j^{\|} = \mathbf{P}_{\mathbf{A}_j}^{\mathcal{Z}_{j-1}}\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right) = \mathbf{Z}_{j-1}\boldsymbol{\eta}_j^\star$ (from Eqs. (1) and (7)) and $\mathbf{x}_j^{\perp} = \left(\mathbf{I} - \mathbf{P}_{\mathbf{A}_j}^{\mathcal{Z}_{j-1}}\right)\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right)$. We can then bound the error as

$$\left\|\mathbf{x}_j^\star - \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right)\right\|_{\mathbf{A}_j} = \left\|\mathbf{x}_j^\star - \bar{\mathbf{x}}_j - \mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}_j}\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right)\right\|_{\mathbf{A}_j} = \left\|\mathbf{x}_j^{\|} + \mathbf{x}_j^{\perp} - \mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}_j}\left(\mathbf{x}_j^{\|}\right)\right\|_{\mathbf{A}_j} \tag{30}$$

$$= \left\|\left(\mathbf{I} - \mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}_j}\right)\left(\mathbf{Z}_{j-1}\boldsymbol{\eta}_j^\star\right)\right\|_{\mathbf{A}_j} + \left\|\mathbf{x}_j^{\perp}\right\|_{\mathbf{A}_j} \tag{31}$$

$$\leq z_{j-1}^{1/2}\sqrt{\sum_{i=1}^{z_{j-1}}\left\|\left(\mathbf{I} - \mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}_j}\right)\left(\eta_i^\star\mathbf{z}_i\right)\right\|_{\mathbf{A}_j}^2} + \underbrace{\left\|\mathbf{x}_j^{\perp}\right\|_{\mathbf{A}_j}}_{\text{(I)}}, \tag{32}$$

where we have used $\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}_j}\left(\mathbf{x}_j^{\perp}\right) = 0$ (because $\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Z}_{j-1}}\left(\mathbf{x}_j^{\perp}\right) = 0$ and $\mathcal{Y}_j \subseteq \mathcal{Z}_{j-1}$), the triangle inquality, and the norm-equivalence relation $\|\mathbf{x}\|_1 \leq n^{1/2}\|\mathbf{x}\|_2$. By comparing Eqs. (25) and (32), using monotinicity of the square root function, and noting that term (I) is independent of the subspace $\mathcal{Y}_j$, it is clear that the POD subspace minimizes (over all y-dimensional subspaces of $\mathcal{Z}_{j-1}$) an upper bound for $\left\|\mathbf{x}_j^\star - \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right)\right\|_{\mathbf{A}_j}$ if $\mathbf{\Theta} = \mathbf{A}_j$, $\mathbf{w}_i = \mathbf{z}_i$ and $\gamma_i = \eta_i^\star$ for $i = 1, \ldots, s$ with $s = z_{j-1}$. □

We now show that other ingredients can be selected to align POD with minimizing $\|\mathbf{z}\left(\mathbf{x}_j^\star\right) - \mathbf{z}\left(\mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right)\right)\|_2$, which is the error minimized by ideal subspace $\bar{\mathcal{Y}}_j^\star$ in problem (24).

Theorem 2. *The POD subspace* $\mathcal{U}_y^{\mathbf{C}^T\mathbf{C}}\left(\mathbf{Z}_{j-1}, \bar{\boldsymbol{\eta}}_j^\star\right)$ *with*

$$\bar{\boldsymbol{\eta}}_j^\star := \left(\mathbf{C}\mathbf{Z}_{j-1}\right)^+ \mathbf{C}\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right) \tag{33}$$

minimizes an upper bound for the output error $\|\mathbf{z}\left(\mathbf{x}_j^\star\right) - \mathbf{z}\left(\mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right)\right)\|_2$ *over all* y*-dimensional subspaces of* $\mathcal{Z}_{j-1} \subseteq \mathbb{R}^n$*, where* $y \leq z_{j-1} \leq n$.

Proof. We again decompose the centered exact solution $\mathbf{x}_j^\star - \bar{\mathbf{x}}_j$ according to

$$\mathbf{x}_j^\star - \bar{\mathbf{x}}_j = \mathbf{x}_j^C + \mathbf{x}_j^{C^\perp} = \underline{\mathbf{x}_j}^{\|} + \underline{\mathbf{x}_j}^{\perp} + \mathbf{x}_j^{C^\perp}. \tag{34}$$

Here, $\mathbf{x}_j^C = \mathbf{P}_{\mathbf{I}}^{\text{range}(\mathbf{C}^T)}\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right)$, $\mathbf{x}_j^{C^\perp} = \left(\mathbf{I} - \mathbf{P}_{\mathbf{I}}^{\text{range}(\mathbf{C}^T)}\right)\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right)$, $\underline{\mathbf{x}_j}^{\|} = \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Z}_{j-1}}\left(\mathbf{x}_j^C\right) = \mathbf{Z}_{j-1}\bar{\boldsymbol{\eta}}_j^\star$ (from Eq. (8)), and $\underline{\mathbf{x}_j}^{\perp} = \left(\mathbf{I} - \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Z}_{j-1}}\right)\left(\mathbf{x}_j^C\right)$. We can then bound the error as

$$\|\mathbf{z}\left(\mathbf{x}_j^\star\right) - \mathbf{z}\left(\mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right)\right)\|_2 = \|\mathbf{C}(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j) - \mathbf{C}(\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}_j}\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right))\|_2 = \left\|\mathbf{x}_j^\star - \bar{\mathbf{x}}_j - \mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}_j}\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right)\right\|_{\mathbf{C}^T\mathbf{C}} \tag{35}$$

$$\leq \left\|\mathbf{x}_j^\star - \bar{\mathbf{x}}_j - \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Y}_j}\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right)\right\|_{\mathbf{C}^T\mathbf{C}} + \left\|(\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}_j} - \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Y}_j})(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j)\right\|_{\mathbf{C}^T\mathbf{C}} \tag{36}$$

$$\leq \left\|\mathbf{x}_j^\star - \bar{\mathbf{x}}_j - \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Y}_j}\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right)\right\|_{\mathbf{C}^T\mathbf{C}} + \left\|\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}_j} - \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Y}_j}\right\|_{\mathbf{C}^T\mathbf{C}} \left\|\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right\|_{\mathbf{C}^T\mathbf{C}} \tag{37}$$

$$= \left\|\underline{\mathbf{x}_j}^{\|} + \underline{\mathbf{x}_j}^{\perp} - \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Y}_j}\left(\underline{\mathbf{x}_j}^{\|}\right)\right\|_{\mathbf{C}^T\mathbf{C}} + \left\|\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}_j} - \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Y}_j}\right\|_{\mathbf{C}^T\mathbf{C}} \left\|\underline{\mathbf{x}_j}^{\|} + \underline{\mathbf{x}_j}^{\perp}\right\|_{\mathbf{C}^T\mathbf{C}} \tag{38}$$

$$= \left\|\left(\mathbf{I} - \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Y}_j}\right)\left(\underline{\mathbf{x}_j}^{\|}\right)\right\|_{\mathbf{C}^T\mathbf{C}} + \left\|\underline{\mathbf{x}_j}^{\perp}\right\|_{\mathbf{C}^T\mathbf{C}} + \left\|\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}_j} - \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Y}_j}\right\|_{\mathbf{C}^T\mathbf{C}} \left\|\underline{\mathbf{x}_j}^{\|} + \underline{\mathbf{x}_j}^{\perp}\right\|_{\mathbf{C}^T\mathbf{C}} \tag{39}$$

$$\leq z_{j-1}^{1/2}\sqrt{\sum_{i=1}^{z_{j-1}} \left\|\left(\mathbf{I} - \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Y}_j}\right)\left([\bar{\boldsymbol{\eta}}_j^\star]_i \mathbf{z}_i\right)\right\|_{\mathbf{C}^T\mathbf{C}}^2} + \underbrace{\left\|\underline{\mathbf{x}_j}^{\perp}\right\|_{\mathbf{C}^T\mathbf{C}} + \alpha\left\|\mathbf{Z}_{j-1}\bar{\boldsymbol{\eta}}_j^\star\right\|_{\mathbf{C}^T\mathbf{C}}}_{\text{(I)}} \tag{40}$$

Here, we have used $\mathbf{C}\mathbf{x}_j^{C^\perp} = 0$ and $\mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Y}_j}\left(\underline{\mathbf{x}_j}^{\perp}\right) = 0$. Also, $\|\mathbf{A}\|_{\mathbf{C}^T\mathbf{C}} = \sup_{\mathbf{x}\neq 0}\|\mathbf{A}\mathbf{x}\|_{\mathbf{C}^T\mathbf{C}} / \|\mathbf{x}\|_{\mathbf{C}^T\mathbf{C}}$ with $\mathbf{A} \in \mathbb{R}^{n\times n}$ is an induced matrix norm and $\alpha := \sup_{\mathcal{Y}\in\mathcal{G}(y,n)}\left\|\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Y}} - \mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Y}}\right\|_{\mathbf{C}^T\mathbf{C}}$. Within inequality (40), term (I) is independent of the subspace $\mathcal{Y}$; as such, a comparison of Eqs. (25) and (40) reveals that the POD subspace minimizes (over all subspaces $\mathcal{Y}$) an upper bound for $\|\mathbf{z}\left(\mathbf{x}_j^\star\right) - \mathbf{z}\left(\mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right)\right)\|_2$ if $\mathbf{\Theta} = \mathbf{C}^T\mathbf{C}$, $\mathbf{w}_i = \mathbf{z}_i$ and $\gamma_i = [\bar{\boldsymbol{\eta}}_j^\star]_i$ for $i = 1, \ldots, s$ with $s = z_{j-1}$ under the stated assumptions. □

3.3 Goal-oriented POD ingredients

In light of these theoretical results, we now propose several practical choices for POD ingredients that align goal-oriented POD with augmented CG.

3.3.1 Snapshots

Theorems 1 and 2 show that POD minimizes an upper bound for errors of interest if snapshots are set to vectors accumulated over the first $j-1$ linear solves. We therefore employ $\mathbf{W} = \mathbf{Z}_{j-1}$ in Eq. (28).

3.3.2 Weights

In practice, the augmenting subspace $\mathcal{Y}_j$ is computed after linear system $j-1$ is solved. As such, we cannot compute the 'ideal weights' $\bar{\boldsymbol{\eta}}_j^\star$ defined in (33), which requires knowledge of $\mathbf{x}_j^\star$. While $\boldsymbol{\eta}_j^\star$ in Eq. (29) can be computed by solving $\mathbf{Z}_{j-1}^T\mathbf{A}_j\mathbf{Z}_{j-1}\boldsymbol{\eta}_j^\star = \mathbf{Z}_{j-1}^T(\mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j)$, this is not practical: the computational cost of doing so is equivalent to solving Eqs. (13) with $\mathbf{Y}_j = \mathbf{Z}_{j-1}$, i.e., employing an augmenting subspace that has not been truncated. As such, we consider two approximations to these ideal weights.

1. **Previous weights**. This approach employs weights of

$$\boldsymbol{\eta}_j^{\text{prev}} = \left(\mathbf{Z}_{j-1}^T\mathbf{A}_{j-1}\mathbf{Z}_{j-1}\right)^{-1}\mathbf{Z}_{j-1}^T\left(\mathbf{b}_{j-1} - \mathbf{A}_{j-1}\bar{\mathbf{x}}_{j-1}\right). \tag{41}$$

Comparing Eqs. (29) and (41) reveals that $\boldsymbol{\eta}_j^{\text{prev}}$ is 'close' to $\boldsymbol{\eta}_j^\star$, but employs readily available data, as these weights are equal to the coefficient in the expansion of the solution at the previous time step, i.e., $\mathbf{x}_{j-1} = \bar{\mathbf{x}}_{j-1} + \mathbf{Z}_{j-1}\boldsymbol{\eta}_j^{\text{prev}}$. We now provide bounds for the difference between these computable weights and the ideal weights.

Theorem 3. *The difference between the previous weights and ideal weights can be bounded as*

$$\|\boldsymbol{\eta}_j^\star - \boldsymbol{\eta}_j^{prev}\| \leq \frac{1}{\sigma_{min}}\left(\|\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Z}_{j-1}} - \mathbf{P}_{\mathbf{A}_{j-1}}^{\mathcal{Z}_{j-1}}\|\|\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\| + \sigma_1\|(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j) - (\mathbf{x}_{j-1}^\star - \bar{\mathbf{x}}_{j-1})\|\right) \tag{42}$$

and

$$\|\bar{\boldsymbol{\eta}}_j^\star - \boldsymbol{\eta}_j^{prev}\| \leq \frac{1}{\sigma_{min}}\left(\|\mathbf{P}_{\mathbf{C}^T\mathbf{C}}^{\mathcal{Z}_{j-1}} - \mathbf{P}_{\mathbf{A}_{j-1}}^{\mathcal{Z}_{j-1}}\|\|\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\| + \sigma_1\|(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j) - (\mathbf{x}_{j-1}^\star - \bar{\mathbf{x}}_{j-1})\|\right), \tag{43}$$

where σ_{min} denotes the smallest singular value of $\mathbf{Z}_{j-1}$ and σ_1 denotes the largest singular value of $\mathbf{P}_{\mathbf{A}_{j-1}}^{\mathcal{Z}_{j-1}}$.

Proof. We prove the result in Eq. (42); Eq. (43) follows trivially. First note that $\mathbf{Z}_{j-1}\left(\boldsymbol{\eta}_j^\star - \boldsymbol{\eta}_j^{\text{prev}}\right) = \mathbf{P}_{\mathbf{A}_j}^{\mathcal{Z}_{j-1}}(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j) - \mathbf{P}_{\mathbf{A}_{j-1}}^{\mathcal{Z}_{j-1}}(\mathbf{x}_{j-1}^\star - \bar{\mathbf{x}}_{j-1})$ from Eqs. (7), (29), and (41). Then, we have

$$\begin{aligned}\|\mathbf{Z}_{j-1}\left(\boldsymbol{\eta}_j^\star - \boldsymbol{\eta}_j^{\text{prev}}\right)\| \leq& \|\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Z}_{j-1}}(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j) - \mathbf{P}_{\mathbf{A}_{j-1}}^{\mathcal{Z}_{j-1}}(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j)\| + \|\mathbf{P}_{\mathbf{A}_{j-1}}^{\mathcal{Z}_{j-1}}\left(\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right) - \left(\mathbf{x}_{j-1}^\star - \bar{\mathbf{x}}_{j-1}\right)\right)\| \\ \leq& \|\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Z}_{j-1}} - \mathbf{P}_{\mathbf{A}_{j-1}}^{\mathcal{Z}_{j-1}}\|\|\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\| + \sigma_1\|(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j) - (\mathbf{x}_{j-1}^\star - \bar{\mathbf{x}}_{j-1})\|\end{aligned}$$

where we have applied the triangle inequality and used $\sigma_1 = \max_{\mathbf{x}\neq 0}\|\mathbf{P}_{\mathbf{A}_{j-1}}^{\mathcal{Z}_{j-1}}\mathbf{x}\|/\|\mathbf{x}\|$. Eq. (42) follows from applying $\sigma_{\min} = \min_{\mathbf{x}\neq 0}\|\mathbf{Z}_{j-1}\mathbf{x}\|/\|\mathbf{x}\|$. □

We now provide conditions under which the computable weights are equal to the ideal weights.

Corollary 1. *If $\mathbf{A}_{j-1} = \mathbf{A}_j$ and $\mathbf{x}_j^\star - \bar{\mathbf{x}}_j = \mathbf{x}_{j-1}^\star - \bar{\mathbf{x}}_{j-1}$, then $\boldsymbol{\eta}_j^\star = \boldsymbol{\eta}_j^{prev}$. Alternatively, if $\mathbf{C}^T\mathbf{C} = \mathbf{A}_j$ and $\mathbf{x}_j^\star - \bar{\mathbf{x}}_j = \mathbf{x}_{j-1}^\star - \bar{\mathbf{x}}_{j-1}$, then $\bar{\boldsymbol{\eta}}_j^\star - \boldsymbol{\eta}_j^{prev}$.*

Proof. The result follows trivially from Eqs. (42) and (43). □

2. **Radial basis functions**. Noting that $\mathbf{Z}_{j-1}\boldsymbol{\eta}_j^\star = \mathbf{P}_{\mathbf{A}_j}^{\mathcal{Z}_{j-1}}\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right)$, we can approximate ideal weights $\boldsymbol{\eta}_j^\star$ by approximating the component of $\mathbf{x}_j^\star - \bar{\mathbf{x}}_j$ in $\mathcal{Z}_{j-1}$. To achieve this, we assume the jth solution can be approximated as a linear combination of previous solutions, i.e.,

$$\mathbf{P}_{\mathbf{A}_j}^{\mathcal{Z}_{j-1}}\left(\mathbf{x}_j^\star - \bar{\mathbf{x}}_j\right) \approx \sum_{i=1}^{\omega}\rho^{\text{RBF}}(j, j-i)(\mathbf{P}_{\mathbf{A}_{j-i}}^{\mathcal{Z}_{j-i}}\left(\mathbf{x}_{j-i}^\star - \bar{\mathbf{x}}_{j-i}\right)), \tag{44}$$

where $\omega \in \mathbb{N}(j-1)$ denotes the number of previous solutions to include and $\rho^{\mathrm{RBF}}(i,j)$ denotes a radial basis function. This implies weights of

$$\begin{aligned}\boldsymbol{\eta}_j^{\mathrm{RBF}}(\omega) &= \sum_{i=1}^{\omega} \rho^{\mathrm{RBF}}(j,j-i)\left(\mathbf{Z}_{j-i}^T\mathbf{A}_{j-i}\mathbf{Z}_{j-i}\right)^{-1}\mathbf{Z}_{j-i}^T\mathbf{A}_{j-i}\left(\mathbf{x}_{j-i}^\star - \bar{\mathbf{x}}_{j-i}\right)\\ &= \sum_{i=1}^{\omega} \rho^{\mathrm{RBF}}(j,j-i)\boldsymbol{\eta}_{j+1-i}^{\mathrm{prev}}.\end{aligned} \tag{45}$$

In practice, we employ a inverse-distance-weight radial basis function of the form $\rho^{\mathrm{RBF}}(i,j) = \rho^{\mathrm{IDW}}(|i-j|)$ with $\rho^{\mathrm{IDW}}(r) := 1/r^2$; we set ω to be the number of linear systems since the most recent truncation.

3.3.3 Metric

Theorems 1 and 2 demonstrate that POD minimizes an upper bound for the errors $\left\|\mathbf{x}_j^\star - \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right)\right\|_{\mathbf{A}_j}$ and $\|\mathbf{z}\left(\mathbf{x}_j^\star\right) - \mathbf{z}\left(\mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right)\right)\|_2$ only if POD metrics of $\boldsymbol{\Theta} = \mathbf{A}_j$ and $\boldsymbol{\Theta} = \mathbf{C}^T\mathbf{C}$ are used, respectively. However, as previously discussed, we aim to avoid using $\mathbf{A}_j$ to truncate the basis $\mathbf{Z}_{j-1}$, as it entails reduced computations with the jth linear system (i.e., solving Eqs. (13) with $\mathbf{Y}_j = \mathbf{Z}_{j-1}$). Therefore, we employ two practical choices for metrics in Eq. (28):

1. $\boldsymbol{\Theta} = \mathbf{A}_{j-1}$. This approach aligns the truncation with minimizing $\left\|\mathbf{x}_j^\star - \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right)\right\|_{\mathbf{A}_j}$. From Eq. (27), the resulting basis is $\mathbf{A}_{j-1}$-orthogonal. Here, computing the POD basis via Algorithm 4 is appropriate, as a symmetric factorization of $\mathbf{A}_{j-1}$ is not readily available.
2. $\boldsymbol{\Theta} = \mathbf{C}^T\mathbf{C}$. This is an output-oriented approach associated with minimizing the output error. From Eq. (27), the resulting basis is $\mathbf{C}^T\mathbf{C}$-orthogonal. Computing this POD basis via Algorithm 5 is appropriate, as a symmetric factor of the (pseudo)metric is readily available as $\mathbf{C}$.

3.3.4 Summary

In summary, the four goal-oriented POD truncation methods we propose are

1. $\mathbf{Y}_{j+1} = \mathbf{U}_{y_{j+1}}^{\mathbf{A}_j}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{prev}}\right)$, computable by $(\mathbf{Y}_{j+1}) = \texttt{pod\_evd}(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{prev}}, \mathbf{A}_j, \varepsilon_y)$,
2. $\mathbf{Y}_{j+1} = \mathbf{U}_{y_{j+1}}^{\mathbf{A}_j}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(\omega)\right)$, computable by $(\mathbf{Y}_{j+1}) = \texttt{pod\_evd}(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(\omega), \mathbf{A}_j, \varepsilon_y)$,
3. $\mathbf{Y}_{j+1} = \mathbf{U}_{y_{j+1}}^{\mathbf{C}^T\mathbf{C}}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{prev}}\right)$, computable by $(\mathbf{Y}_{j+1}) = \texttt{pod\_svd}(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{prev}}, \mathbf{C}, \varepsilon_y)$,
4. $\mathbf{Y}_{j+1} = \mathbf{U}_{y_{j+1}}^{\mathbf{C}^T\mathbf{C}}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(\omega)\right)$, computable by $(\mathbf{Y}_{j+1}) = \texttt{pod\_svd}(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(\omega), \mathbf{C}, \varepsilon_y)$,

where $\varepsilon_y \in [0,1]$ is a statistical 'energy criterion' used to truncate the POD basis, and `pod_evd` and `pod_svd` are described in Algorithms 4 and 5, respectively.

4 Three-stage algorithm

We propose a novel three-stage algorithm for augmented PCG to efficiently compute inexact solutions; the algorithm leverages an augmenting basis that is optimally ordered and yields a well-conditioned reduced matrix. In particular, for linear system j, we assume that an augmenting basis $\mathbf{Y}_j \in \mathbb{R}_\star^{n\times y_j}$ is available that (1) contains a low-dimensional basis $\mathbf{W}_j \in \mathbb{R}_\star^{n\times w_j}$ with $\mathcal{W}_j \subseteq \mathcal{Y}_j \subseteq \mathbb{R}^n$ and $w_j \leq y_j \leq n$ that can capture an accurate solution, and (2) yields a well-conditioned reduced matrix $\mathbf{Y}_j^T\mathbf{A}_j\mathbf{Y}_j$.

Goal-oriented POD basis fits naturally into this framework. First, it is optimally ordered (Remark 1): the first few POD basis vectors span an optimal subspace in the sense of minimizing the objective function in Eq. (25), which is an an upper bound for the $\mathbf{A}_j$-norm and $\mathbf{C}^T\mathbf{C}$-norm of the error, respectively (Theorems 1 and 2). This implies that the first few POD vectors could be employed as $\mathbf{W}_j$. Second, it automatically yields a well-conditioned reduced matrix if $\boldsymbol{\Theta} = \mathbf{A}_j$; this will be further discussed in Section 4.4.1. However, other truncation methods that satisfy these properties can also be considered within the proposed three-stage algorithm.

4.1 Stage 1

The objective of Stage 1 is to 'jump start' the algorithm by computing an accurate solution at very low cost. To achieve this, this stage computes $\bar{\mathbf{x}}_j + \mathbf{W}_j\hat{\mathbf{w}}_j = \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{W}_j}\left(\mathbf{x}_j^\star\right)$ by solving

$$\mathbf{W}_j^T\mathbf{A}_j\mathbf{W}_j\hat{\mathbf{w}}_j = \mathbf{W}_j^T(\mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j) \tag{46}$$

directly. The assumptions placed on $\mathbf{W}_j$ imply that the solution $\bar{\mathbf{x}}_j + \mathbf{W}_j\hat{\mathbf{w}}_j$ will be accurate and a direct solve will be inexpensive. Computing this solution can be executed via Algorithm 2 as

$$(\hat{\mathbf{w}}_j, \hat{\mathbf{R}}_j) = \texttt{direct\_reduced\_solve}(\mathbf{A}_j, \mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j, \mathbf{W}_j).$$

Algorithm 2 `direct_reduced_solve`

Input: $\mathbf{A}$, $\mathbf{b}$, $\mathbf{W}$
Output: $\hat{\mathbf{w}}$, $\hat{\mathbf{R}}$
1: Compute $\hat{\mathbf{A}} = \mathbf{W}^T\mathbf{A}\mathbf{W}$ and $\hat{\mathbf{b}} = \mathbf{W}^T\mathbf{b}$
2: Solve $\hat{\mathbf{A}}\hat{\mathbf{w}} = \hat{\mathbf{b}}$ by Cholesky factorization $\hat{\mathbf{A}} = \hat{\mathbf{R}}^T\hat{\mathbf{R}}$

4.2 Stage 2

The objective of Stage 2 is to improve upon the stage-1 solution $\bar{\mathbf{x}}_j + \mathbf{W}_j\hat{\mathbf{w}}_j$ by efficiently solving over the entire augmenting subspace $\mathcal{Y}_j$, whose dimension y_j may be modest. To achieve this, this stage solves

$$\mathbf{Y}_j^T\mathbf{A}_j\mathbf{Y}_j\hat{\mathbf{y}}_j^\star = \mathbf{Y}_j^T\left(\mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j - \mathbf{A}_j\mathbf{W}_j\hat{\mathbf{w}}_j\right) \tag{47}$$

iteratively to tolerance $\hat{\epsilon}_j$ via augmented CG with augmenting basis $\hat{\mathbf{W}}_j \in \mathbb{R}_\star^{y_j\times w_j}$, which represents the stage-1 basis in augmenting-subspace coordinates, i.e., $\mathbf{W}_j = \mathbf{Y}_j\hat{\mathbf{W}}_j$. This approach is promising for two reasons. First, it precludes the need to explicitly compute the reduced matrix $\mathbf{Y}_j^T\mathbf{A}_j\mathbf{Y}_j$, which consumes $\mathcal{O}((\tau_j+y_j)y_jn)$ flops, where τ_j denotes the average number of nonzeros per row of $\mathbf{A}_j$. Instead, matrix–vector products of the form $\mathbf{Y}_j^T\mathbf{A}_j\mathbf{Y}_j\mathbf{p}$ can be computed via $\mathbf{Y}_j^T(\mathbf{A}_j(\mathbf{Y}_j\mathbf{p}))$ in $\mathcal{O}((2y_j+\tau_j)n)$ flops. Second, a small number of iterations will be needed for convergence if the reduced matrix $\mathbf{Y}_j^T\mathbf{A}_j\mathbf{Y}_j$ is well conditioned, as has been assumed.

Therefore, this stage amounts to executing Algorithm 1 as

$$(\hat{k}_j, \hat{\mathbf{x}}_j, \hat{\mathbf{X}}_j, \hat{\mathbf{\Gamma}}_j) = \texttt{augmented\_pcg}(\mathbf{Y}_j^T\mathbf{A}_j\mathbf{Y}_j, \mathbf{Y}_j^T(\mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j), \hat{\mathbf{W}}_j\hat{\mathbf{w}}_j, \hat{\mathbf{W}}_j, \mathbf{I}, \hat{\epsilon}_j)$$

with two implementation optimizations:

- Matrix–vector products of the form $\mathbf{Y}^T\mathbf{A}\mathbf{Y}\mathbf{p}$ appearing in steps 4, 10, and 13 of Algorithm 1 can be computed efficiently via $\mathbf{Y}^T(\mathbf{A}(\mathbf{Y}\mathbf{p}))$ as discussed above.
- The solves in steps 4 and 13 of Algorithm 1 can be performed directly in $\mathcal{O}(w_j^2)$ flops, as the Cholesky factorization of $\mathbf{Y}^T\mathbf{A}\mathbf{Y} = \hat{\mathbf{R}}_j^T\hat{\mathbf{R}}_j$ can be reused from stage 1.

The (inexact) solution increment $\mathbf{Y}_j\hat{\mathbf{y}}_j = \mathbf{X}_j\hat{\mathbf{x}}_j$ computed by stage 2 lies in the range of a (reduced) Krylov basis $\mathbf{X}_j = \mathbf{Y}_j\hat{\mathbf{X}}_j \in \mathbb{R}^{n\times\hat{k}_j}$ with $\mathcal{X}_j \subseteq \mathcal{Y}_j \subseteq \mathbb{R}^n$ and $\hat{\mathbf{X}}_j \in \mathbb{R}^{y_j\times\hat{k}_j}$, where $\hat{k}_j \leq y_j$ denotes the number of stage-2 iterations. This basis satisfies $\mathbf{X}_j^T\mathbf{A}_j\mathbf{X}_j = \hat{\mathbf{\Gamma}}_j$ and $\mathbf{X}_j^T\mathbf{A}_j\mathbf{W}_j = 0$, while the resulting solution satisfies

$$\bar{\mathbf{x}}_j + \mathbf{W}_j\hat{\mathbf{w}}_j + \mathbf{X}_j\hat{\mathbf{x}}_j = \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{W}_j+\mathcal{X}_j}\left(\mathbf{x}_j^\star\right) \tag{48}$$

Ref. [20] proposed a similar idea referred to as the iterative reuse of Krylov subspaces (IRKS). This approach did not employ stage-1 direct solve and suppored using either $\mathbf{Y}_j = [\mathbf{V}_1\ \cdots\ \mathbf{V}_{j-1}]$ or $\mathbf{Y}_j = \mathbf{V}_{j-1}$ (i.e., no truncation).

4.3 Stage 3

The objective of stage 3 is to continue iterating in the full space until a specified tolerance ϵ_j is satisfied. This stage therefore solves

$$\mathbf{A}_j \delta\mathbf{x}_j^\star = \mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j - \mathbf{A}_j\mathbf{W}_j\hat{\mathbf{w}}_j - \mathbf{A}_j\mathbf{X}_j\hat{\mathbf{x}}_j \tag{49}$$

via augmented PCG with augmenting basis $[\mathbf{W}_j,\ \mathbf{X}_j]$. This amounts to executing Algorithm 1 as

$$(k_j, \hat{\mathbf{v}}_j, \mathbf{V}_j, \mathbf{\Gamma}_j) = \texttt{augmented\_pcg}(\mathbf{A}_j, \mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j, \left[\hat{\mathbf{w}}_j^T,\ \hat{\mathbf{x}}_j^T\right]^T, [\mathbf{W}_j,\ \mathbf{X}_j], \mathbf{M}_j, \epsilon_j)$$

with the following optimization:

- The solves in steps 4 and 13 can be performed directly in $\mathcal{O}(w_j^2 + \hat{k}_j)$ flops, as the Cholesky factorization $\mathbf{Y}^T\mathbf{A}\mathbf{Y} = \mathbf{R}_j^T\mathbf{R}_j$ is readily available from stages 1 and 2 as

$$\mathbf{R}_j = \begin{bmatrix} \hat{\mathbf{R}}_j & \mathbf{0} \\ \mathbf{0} & \sqrt{\hat{\mathbf{\Gamma}}_j} \end{bmatrix}.$$

The solution increment $\delta\mathbf{x}_j = \mathbf{V}_j\hat{\mathbf{v}}_j$ computed by stage 3 lies in the range of a Krylov basis $\mathbf{V}_j \in \mathbb{R}_\star^{n\times k_j}$, where $k_j \leq n$ denotes the number of stage-3 iterations. This basis satisfies $\mathbf{V}_j^T\mathbf{A}_j\mathbf{V}_j = \mathbf{\Gamma}_j$ with $\mathbf{\Gamma}_j \in \mathbb{R}^{k_j\times k_j}$ diagonal and $\mathbf{V}_j^T\mathbf{A}_j[\mathbf{W}_j,\ \mathbf{X}_j] = \mathbf{0}$, while the resulting solution satisfies

$$\bar{\mathbf{x}}_j + \mathbf{W}_j\hat{\mathbf{w}}_j + \mathbf{X}_j\hat{\mathbf{x}}_j + \mathbf{V}_j\hat{\mathbf{v}}_j = \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{W}_j+\mathcal{X}_j+\mathcal{V}_j}\left(\mathbf{x}_j^\star\right). \tag{50}$$

4.3.1 Accounting for entire augmenting subspace

The stage-3 implementation described above augments with the subspace range$([\mathbf{W}_j,\ \mathbf{X}_j]) \subseteq \mathcal{Y}_j$. However, if the stage-2 tolerance $\hat{\epsilon}_j$ is small, then we know the stage-2 solution $\bar{\mathbf{x}}_j + \mathbf{W}_j\hat{\mathbf{w}}_j + \mathbf{X}_j\hat{\mathbf{x}}_j$ is *close* to being optimal over the entire augmenting subspace, i.e.,

$$\bar{\mathbf{x}}_j + \mathbf{W}_j\hat{\mathbf{w}}_j + \mathbf{X}_j\hat{\mathbf{x}}_j = \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{W}_j+\mathcal{X}_j}\left(\mathbf{x}_j^\star\right) \approx \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right).$$

Therefore, employing $\mathbf{Y} \leftarrow \mathbf{Y}_j$ in stage 3 may reduce the number of iterations to convergence, as this ensures that new search directions remain $\mathbf{A}_j$-orthogonal to the full augmenting subspace $\mathcal{Y}_j$ over which the solution has already been computed to tolerance $\hat{\epsilon}_j$.

However, for this approach to be practical, the reduced solves in steps 4 and 13 must be performed efficiently; performing these solves directly requires computing and factorizing the matrix $\mathbf{Y}_j^T\mathbf{A}_j\mathbf{Y}_j$, which is not available from stages 1 or 2. In fact, computing and factorizing this matrix incurs $\mathcal{O}((\tau_j + {y_j}^2)n + {y_j}^3)$ flops: the same computational cost as executing stage 1 with $\mathbf{W}_j = \mathbf{Y}_j$. We therefore propose performing these solves *iteratively*, i.e, by executing stage 2 within stage 3. In particular, this stage-3 variant executes Algorithm 1 as

$$(k_j, \hat{\mathbf{v}}_j, \mathbf{V}_j, \mathbf{\Gamma}_j) = \texttt{augmented\_pcg}(\mathbf{A}_j, \mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j, \hat{\mathbf{W}}_j\hat{\mathbf{w}}_j + \hat{\mathbf{X}}_j\hat{\mathbf{x}}_j, \mathbf{Y}_j, \mathbf{M}_j, \epsilon_j)$$

with the following optimizations:

- At iteration k, the solves in steps 4 and 13 can be performed iteratively by executing Algorithm 1 as

  $(\bar{k}_j^{(k)}\ \hat{\mathbf{x}}_j^{(k)}\ \bar{\mathbf{X}}_j^{(k)}\ \bar{\mathbf{\Gamma}}_j^{(k)}) = \texttt{augmented\_pcg}(\mathbf{Y}_j^T\mathbf{A}_j\mathbf{Y}_j, \mathbf{Y}_j^T(\mathbf{A}_j\mathbf{z}_j^{(k+1)}), \mathbf{0}, [\hat{\mathbf{W}}_j,\ \hat{\mathbf{X}}_j,\ \bar{\mathbf{X}}_j^{(0)},\ \cdots,\ \bar{\mathbf{X}}_j^{(k-1)}], \mathbf{I}, \bar{\epsilon}_j)$

  with the following optimizations:
  - Matrix–vector products of the form $\mathbf{Y}^T\mathbf{A}\mathbf{Y}\mathbf{p}$ appearing in steps 4, 10, and 13 can be computed efficiently via $\mathbf{Y}^T(\mathbf{A}(\mathbf{Y}\mathbf{p}))$.
  - The solves in steps 4 and 13 can be performed directly in $\mathcal{O}(w_j^2 + \hat{k}_j + \sum_{\ell=0}^{k-1}\bar{k}_j^{(\ell)})$ flops, as the Cholesky factorization $\mathbf{Y}^T\mathbf{A}\mathbf{Y} = \bar{\mathbf{R}}_j^{(k)}\bar{\mathbf{R}}_j^{(k)}$ with

$$\bar{\mathbf{R}}_j^{(k)} = \begin{bmatrix} \hat{\mathbf{R}}_j & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \sqrt{\hat{\mathbf{\Gamma}}_j} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sqrt{\bar{\mathbf{\Gamma}}_j^{(0)}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \sqrt{\bar{\mathbf{\Gamma}}_j^{(k-1)}} \end{bmatrix}$$

can be reused from stage 1, stage 2, and earlier stage-3 iterations.

This solution increment $\delta\mathbf{x}_j = \mathbf{V}_j\hat{\mathbf{v}}_j$ also lies in the range of a Krylov basis $\mathbf{V}_j \in \mathbb{R}_\star^{n\times k_j}$ with $\mathbf{V}_j^T\mathbf{A}_j\mathbf{V}_j = \mathbf{\Gamma}_j \in \mathbb{R}^{k_j\times k_j}$ diagonal; however, the basis now satisfies $\mathbf{V}_j^T\mathbf{A}_j[\mathbf{W}_j,\ \mathbf{X}_j,\ \mathbf{Y}_j\bar{\mathbf{X}}_j^{(0)},\ \cdots,\ \mathbf{Y}_j\bar{\mathbf{X}}_j^{(k_j-1)}] = \mathbf{0}$, while the resulting solution satisfies

$$\bar{\mathbf{x}}_j + \mathbf{W}_j\hat{\mathbf{w}}_j + \mathbf{X}_j\hat{\mathbf{x}}_j + \mathbf{V}_j\hat{\mathbf{v}}_j \approx \mathbf{P}_{\mathbf{A}_j}^{\bar{\mathbf{x}}_j+\mathcal{Y}_j}\left(\mathbf{x}_j^\star\right). \tag{51}$$

Note that the solution does not associate with an exact orthogonal projection.

4.4 Overall algorithm

Algorithm 3 reports the proposed three-stage algorithm, where $\bar{y}$ denotes the maximum number of accumulated vectors to preserve before truncation, φ is a boolean variable that is 1 if the entire augmenting subspace (not just stage 1 and 2 directions) should be orthogonalized against as discussed in Section 4.3.1. The variable $\varrho \in [0,1]$ is a threshold for determining which new vectors should be included in the stage 1 basis.

Algorithm 3 `three_stage_algorithm`

Input: $\{\mathbf{A}_j\}_{j=1}^p$, $\{\mathbf{b}_j\}_{j=1}^p$, $\{\bar{\mathbf{x}}_j\}_{j=1}^p$, forcing sequence $\{\epsilon_j\}_{j=1}^p$, storage threshold $\bar{y}$, stage-1 option ϱ, stage-3 option φ
Output:

1: $\mathbf{W}_1 \leftarrow \emptyset$, $\mathbf{X}_1 \leftarrow \emptyset$, $\bar{j} \leftarrow 0$
2: **for** $j = 1,\ldots,p$ **do**
3: **if** $\mathbf{W}_j \neq \emptyset$ **then**
4: *Stage 1*: $(\hat{\mathbf{w}}_j, \hat{\mathbf{R}}_j) =$ `direct_reduced_solve`$(\mathbf{A}_j,\ \mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j,\ \mathbf{W}_j)$
5: **end if**
6: **if** $\mathbf{X}_j \neq \emptyset$ **then**
7: *Stage 2*: $(\hat{k}_j, \hat{\mathbf{x}}_j, \hat{\mathbf{X}}_j, \hat{\mathbf{\Gamma}}_j) =$ `augmented_pcg`$(\mathbf{Y}_j^T\mathbf{A}_j\mathbf{Y}_j,\ \mathbf{Y}_j^T(\mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j),\ \hat{\mathbf{W}}_j\hat{\mathbf{w}}_j,\ \hat{\mathbf{W}}_j,\ \mathbf{I},\ \hat{\epsilon}_j)$.
 Note optimizations discussed in Section 4.2
8: **end if**
9: **if** $\varphi = 1$ **then** {orthogonalize against entire $\mathbf{Y}_j$}
10: *Stage 3*: $(k_j, \hat{\mathbf{v}}_j, \mathbf{V}_j, \mathbf{\Gamma}_j) =$ `augmented_pcg`$(\ \mathbf{A}_j,\ \mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j,\ \hat{\mathbf{W}}_j\hat{\mathbf{w}}_j + \hat{\mathbf{X}}_j\hat{\mathbf{x}}_j,\ \mathbf{Y}_j,\ \mathbf{M}_j,$ and $\epsilon_j)$.
 Note optimizations discussed in Section 4.3.1
11: **else**
12: *Stage 3*: $(k_j, \hat{\mathbf{v}}_j, \mathbf{V}_j, \mathbf{\Gamma}_j) =$ `augmented_pcg`$(\ \mathbf{A}_j,\ \mathbf{b}_j - \mathbf{A}_j\bar{\mathbf{x}}_j,\ \left[\hat{\mathbf{w}}_j^T\ \hat{\mathbf{x}}_j^T\right]^T,\ [\mathbf{W}_j\ \mathbf{X}_j],\ \mathbf{M}_j,\ \epsilon_j)$.
 Note optimizations discussed in Section 4.3
13: **end if**
14: $\mathbf{Y}_{j+1} \leftarrow \left[\mathbf{Y}_j,\ \mathbf{V}_j\sqrt{\mathbf{\Gamma}_j}\right]$
15: $\mathbf{W}_{j+1} \leftarrow \mathbf{W}_j$
16: $\mathcal{K}_j \leftarrow \{i \mid [\mathbf{\Gamma}_j]_{ii} / \sum_{k=1}^{k_j}[\mathbf{\Gamma}_j]_{kk} > \varrho\}$
17: **for** $k \in \mathcal{K}_j$ **do**
18: $\mathbf{W}_{j+1} \leftarrow [\mathbf{W}_{j+1},\ [\mathbf{V}_j]_k\sqrt{[\mathbf{\Gamma}_j]_{kk}}]$
19: **end for**
20: **if** $\varrho = 1$ **then** {include new vectors in stage-1 basis}
21: $\mathbf{W}_{j+1} \leftarrow [\mathbf{W}_j,\ \mathbf{V}_j\sqrt{\mathbf{\Gamma}_j}]$
22: **else**
23: $\mathbf{W}_{j+1} \leftarrow \mathbf{W}_j$
24: **end if**
25: **if** $y_{j+1} > \bar{y}$ **then** {compress}
26: $\mathbf{Y}_{j+1} =$ `compression`$(\mathbf{Y}_{j+1})$
27: Enforce $\mathbf{A}_j$-orthogonality: $\mathbf{Y}_{j+1}^T\mathbf{A}_j\mathbf{Y}_{j+1} = \mathbf{L}_j\mathbf{L}_j^T$ (Cholesky factorization), $\mathbf{Y}_{j+1} \leftarrow \mathbf{Y}_{j+1}\mathbf{L}^{-T}$
28: Determine $\mathbf{W}_{j+1}$ with range$(\mathbf{W}_{j+1}) \subseteq$ range$(\mathbf{Y}_{j+1})$
29: $\bar{j} \leftarrow j$
30: **end if**
31: **end for**

Step 27 of Algorithm 3 ensures that $\mathbf{Y}_{\bar{j}+1}^T\mathbf{A}_{\bar{j}}\mathbf{Y}_{\bar{j}+1} = \mathbf{I}$. We now show that this step is critical for enabling fast stage-2 convergence: it leads to well-conditioned reduced matrices for slowly varying matrix sequences.

Theorem 4. *In Algorithm 3, if either (1)* $\varrho = 1$ *and* $\mathbf{W}_{\bar{j}+1} = \mathbf{Y}_{\bar{j}+1}$ *in step 28, or (2)* $\varphi = 1$ *and* $\hat{\epsilon}_k = \bar{\epsilon}_k = 0$, $k = \bar{j}+1, \ldots, j-1$, *then*

$$\|\mathbf{Y}_j \mathbf{A}_j \mathbf{Y}_j - \mathbf{I}\| \leq \sum_{k=\bar{j}+1}^{j} \|\mathbf{Y}_k\|^2 \|\mathbf{A}_k - \mathbf{A}_{k-1}\|. \tag{52}$$

Proof.

$$\|\mathbf{Y}_j^T \mathbf{A}_j \mathbf{Y}_j - \mathbf{I}\| = \|\mathbf{Y}_j^T \mathbf{A}_{j-1} \mathbf{Y}_j - \mathbf{I} + \mathbf{Y}_j^T (\mathbf{A}_j - \mathbf{A}_{j-1}) \mathbf{Y}_j\| \tag{53}$$

$$\leq \| \left[\mathbf{Y}_{j-1},\ \mathbf{V}_{j-1}\sqrt{\mathbf{\Gamma}_{j-1}}\right]^T \mathbf{A}_{j-1} \left[\mathbf{Y}_{j-1},\ \mathbf{V}_{j-1}\sqrt{\mathbf{\Gamma}_{j-1}}\right] - \mathbf{I}\| + \|\mathbf{Y}_j\|^2 \|\mathbf{A}_j - \mathbf{A}_{j-1}\| \tag{54}$$

$$= \| \begin{bmatrix} \mathbf{Y}_{j-1}\mathbf{A}_{j-1}\mathbf{Y}_{j-1} - \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \| + \|\mathbf{Y}_j\|^2 \|\mathbf{A}_j - \mathbf{A}_{j-1}\|, \tag{55}$$

$$= \|\mathbf{Y}_{j-1}\mathbf{A}_{j-1}\mathbf{Y}_{j-1} - \mathbf{I}\| + \|\mathbf{Y}_j\|^2 \|\mathbf{A}_j - \mathbf{A}_{j-1}\|, \tag{56}$$

$$\tag{57}$$

where we have used $\mathbf{Y}_k^T \mathbf{A}_k \mathbf{V}_k = \mathbf{0}$, $k = 1, \ldots, j-1$ (which holds under the stated assumptions) and $\sqrt{\mathbf{\Gamma}_k}^T \mathbf{V}_k^T \mathbf{A}_k \mathbf{V}_k \sqrt{\mathbf{\Gamma}_k} = \mathbf{I}$, $k = 1, \ldots, j-1$. By induction, we have

$$\|\mathbf{Y}_j^T \mathbf{A}_j \mathbf{Y}_j - \mathbf{I}\| \leq \sum_{k=\bar{j}+1}^{j} \|\mathbf{Y}_k\|^2 \|\mathbf{A}_k - \mathbf{A}_{k-1}\|, \tag{58}$$

where we have used $\mathbf{Y}_{\bar{j}+1}^T \mathbf{A}_{\bar{j}} \mathbf{Y}_{\bar{j}+1} = \mathbf{I}$. □

4.4.1 Integration with goal-oriented POD

Section 3.3.4 described four sets of goal-oriented POD ingredients (with $\omega = j - \bar{j}$ and $\mathbf{Z}_j = \mathbf{Y}_{j+1}$) that could be employed as `compression` in step 26 of Algorithm 3. In all cases, step 28 of Algorithm 3 can be executed by setting $\mathbf{W}_{j+1}$ equal to the first w_j vectors of $\mathbf{Y}_{j+1}$, where w_j is determined from the appropriate POD algorithm (Algorithm 4 or 5) with a modest statistical energy criterion $\varepsilon \leftarrow \varepsilon_w$ with $\varepsilon_w \leq \varepsilon_y$.

The first two options $\mathbf{Y}_{j+1} = \mathbf{U}_{y_{j+1}}^{\mathbf{A}_j}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\text{prev}}\right)$ and $\mathbf{Y}_{j+1} = \mathbf{U}_{y_{j+1}}^{\mathbf{A}_j}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\text{RBF}}(\omega)\right)$ expose two implementation optimizations. First, during any truncation iteration $\bar{j}$, the algorithm should employ $\mathbf{W}_{\bar{j}} = \mathbf{Y}_{\bar{j}}$. The reason is that the compression step requires computing $\mathbf{Y}_{\bar{j}}^T \mathbf{A}_{\bar{j}} \mathbf{Y}_{\bar{j}}$ explicitly; because this is the same matrix used in stage one with $\mathbf{W}_{\bar{j}} = \mathbf{Y}_{\bar{j}}$, employing this choice can lead to faster stage-3 convergence at no additional computational cost. Second, orthogonalization step 27 in Algorithm 3 requires *no operations*, as the basis is automatically $\mathbf{A}_{\bar{j}}$-orthogonal (see Eq. (27)).

5 Numerical experiments

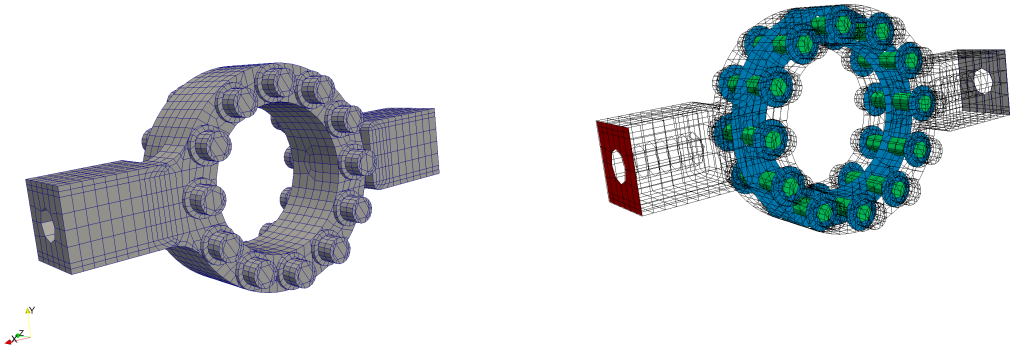
5.1 Problem description

We now assess the proposed methodology using model problems from Sierra/SolidMechanics [13], which is a Lagrangian, three-dimensional code for finite element analysis of solids and structures.

5.1.1 Problem 1: Pancake problem

We first consider computing the quasistatic response of the 'pancake' domain pictured in Figure 1. The material is steel, which is characterized by a Young's modulus of $E = 2.0 \times 10^8 \ \frac{\text{N}}{\text{mm} \cdot \text{s}^2}$, Poisson's ratio of $\nu = 0.3$, density of $\rho = 7.86 \times 10^{-6}$ kg/mm^3. The logarithmic thermal strain of steel is linearly dependent on temperature $\epsilon_{\text{thermal}} = (11.7 \times 10^{-6}) \Delta T \ \frac{1}{\text{K}}$, where ΔT denotes the change in temperature (in Kelvin) from the reference temperature.

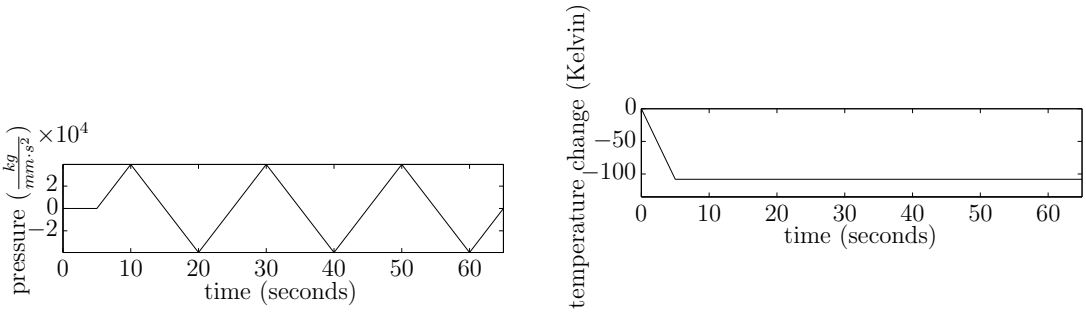
The x-, y-, and z-displacements of the rightmost surface (gray in Figure 1(b)) are set to zero. The x- and y- displacements of the leftmost surface (red in Figure 1(b)) are set to zero; this surface is also subjected to the time-dependent pressure load depicted in Figure 2a.



(a) Finite-element mesh

(b) Pressure-loaded surface (red), contact surfaces (blue), prescribed temperature (green), Dirichlet boundary condition (gray).

Figure 1: Pancake problem.



(a) Time-dependent pressure load applied to leftmost surface (extrema are $\pm 3.94 \times 10^4$ $\frac{\text{kg}}{\text{mm} \cdot \text{s}^2}$).

(b) Time-dependent temperature load applied to bolts.

Figure 2: Time-dependent loadings for the pancake problem.

The time-dependent thermal load depicted in Figure 2b is applied to the bolts (green components in Figure 1(b)) to emulate a pre-loading condition. Contact surfaces are shown in blue in Figure 1(b); they are enforced by an augmented Lagrangian approach with DASH search using a penalty factor of 1.25 and a friction coefficient of 0.5.

The problem is discretized by the finite-element method using a mesh generated by the SIERRA toolkit [7]. The mesh consists of 9108 nodes and 4719 hexahedral elements. At each node, there are three degrees of freedom (the x-, y-, and z-displacements), which leads to $27,324$ total degrees of freedom for the finite-element model.

As the time scales of the load application are relatively large, we neglect inertial effects and consider solving the quasi-static equations

$$\boldsymbol{f}^{\text{int}}\left(\boldsymbol{u}_i\right) + \boldsymbol{f}^{\text{contact}}\left(\boldsymbol{u}_i, \lambda^l\right) = \boldsymbol{f}^{\text{ext}}(t_i), \quad l = 1, \ldots, L, \quad i = 1, \ldots, T, \tag{59}$$

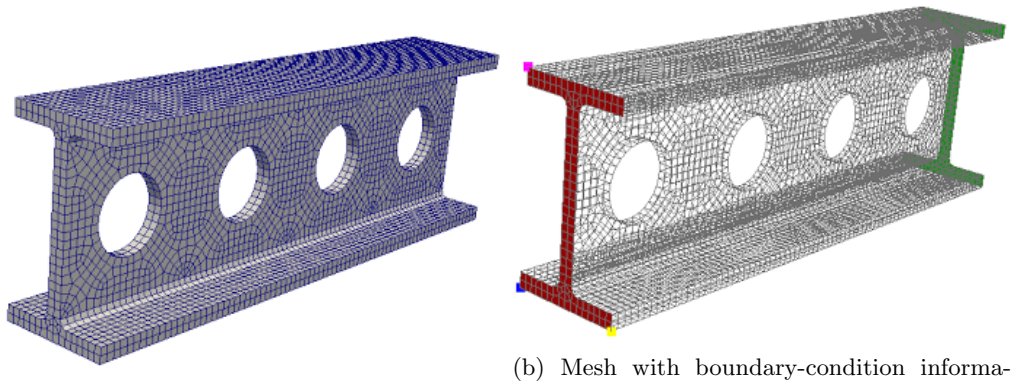
where T denotes the number of time steps, $\boldsymbol{f}^{\text{int}} : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear operator representing the internal forces, $\boldsymbol{f}^{\text{contact}} : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^n$ is nonlinear in its first argument and represents contact forces, $\boldsymbol{f}^{\text{ext}} : [0, 65] \to \mathbb{R}^n$ is the external-force vector, and $\boldsymbol{u}_i \in \mathbb{R}^n$ denotes the displacement at time t_i. As the contact constraints are enforced using an augmented Lagrangian approach within a continuation loop, $\lambda^l \in \mathbb{R}^+$ with $\lambda^l \leq \lambda^{l+1}$, $l \in \mathbb{N}(L)$ denotes the penalty factor at continuation iteration l. Note that these equations also include equality constraints arising from the Dirichlet boundary conditions.

Solving Eqs. (59) is mathematically equivalent to solving

$$\underset{\mathbf{z}\in\mathbb{R}^n}{\text{minimize}} \quad g_i^l(\mathbf{z}) \tag{60}$$

with $-\nabla g_i^l : \mathbf{z} \mapsto \boldsymbol{f}^{\text{ext}}(t_i) - \boldsymbol{f}^{\text{int}}(\mathbf{z}) - \boldsymbol{f}^{\text{contact}}\left(\mathbf{z}, \lambda^l\right)$ for a local minimum. We solve problem (60) using the nonlinear conjugate gradient method with a displacement-dependent preconditioner $\bar{\mathbf{M}}^l : \boldsymbol{u} \mapsto \nabla_{\boldsymbol{u}} \boldsymbol{f}^{\text{int}}(\boldsymbol{u}) + \nabla_{\boldsymbol{u}} \boldsymbol{f}^{\text{contact}}\left(\boldsymbol{u}, \lambda^l\right) \in \text{SPD}(n)$. This results in a sequence of linear systems of the original form in Eqs. (1): one at each nonlinear conjugate-gradient iteration. Here, $\mathbf{A}_j = \bar{\mathbf{M}}^l(\boldsymbol{u}_i^{l(k)})$, $\mathbf{b}_j = \nabla g_i^l(\boldsymbol{u}_i^{l(k)})$, $\boldsymbol{u}_i^{l(k)} \in \mathbb{R}^n$ denotes the displacement at time step i, continuation iteration l, and nonlinear conjugate gradient iteration k. The mapping between timestep i, continuation iteration l, nonlinear conjugate-gradient iteration k and the index of the linear system j is provided by $j : (i, l, k) \mapsto k + \sum_{\mathrm{i}=1}^{i} \sum_{\mathrm{l}=1}^{l} K(\mathrm{i}, \mathrm{l})$, where $K(i, l)$ denotes the number of conjugate-gradient iterations needed to solve problem (60). For this problem, the total number of linear systems we consider is $p = 47$. Each of these linear systems is preconditioned using a three-level algebraic multigrid (AMG) preconditioner $\mathbf{M}_j$ with incomplete Cholesky smoothing for both pre-smoothing and post-smoothing. This preconditioner tends to be expensive to apply, especially when compared with the cost of a matrix–vector product for this system.

5.1.2 Problem 2: I-beam problem



(a) Finite-element mesh

(b) Mesh with boundary-condition information.

Figure 3: I-beam problem.

We next consider computing the quasistatic response (neglecting thermal effects) of the I-beam pictured in Figure 3a, where there are holes in the web section. The material is steel 304L, which is characterized by a Young's modulus of $E = 2.1 \times 10^8 \ \frac{\text{N}}{\text{mm} \cdot \text{s}^2}$, Poisson's ratio of $\nu = 0.33$, and density of $\rho = 7.8 \times 10^{-6}$ kg/mm^3. The x-, y-, and z-displacement of the bottom-left point (blue point in Figure 3b(b)) are set to zero. The x- and y- displacement of the bottom-right point (yellow point in Figure 3b) are set to zero. The x-displacement of the top-left point (magenta point in Figure 3b) is set to zero. Finally, a torsional traction is applied the end surfaces (red and green in Figure 3); Ref. [1, Eq. (11)] reports details on the tractional loading, where the scale factor for the current problem is 0.01. The mesh for this problem consists of 13,137 nodes and 8,576 hexahedral elements. At each node, there are three degrees of freedom (the x-, y-, and z-displacements), which leads to $39,411$ total degrees of freedom for the finite-element model.

We again neglect inertial effects and consider solving the quasi-static equations

$$\boldsymbol{f}^{\text{int}}(\boldsymbol{u}_i) = \boldsymbol{f}^{\text{ext}}(t_i), \quad i = 1, \ldots, T. \tag{61}$$

Note that these equations also include equality constraints arising from the Dirichlet boundary conditions and the continuation loop for computing contact forces does not appear for this problem. Solving Eqs. (61) is equivalent to solving

$$\underset{\mathbf{z}\in\mathbb{R}^n}{\text{minimize}} \quad g_i(\mathbf{z}) \tag{62}$$

with $-\nabla g_i : \mathbf{z} \mapsto \boldsymbol{f}^{\text{ext}}(t_i) - \boldsymbol{f}^{\text{int}}(\mathbf{z})$ for a local minimum. We again use the nonlinear conjugate gradient method to solve problem (62); we also again employ a displacement-dependent preconditioner $\bar{\mathbf{M}}$:

$\boldsymbol{u} \mapsto \nabla_{\boldsymbol{u}} \boldsymbol{f}^{\text{int}}(\boldsymbol{u}) + \nabla_{\boldsymbol{u}} \boldsymbol{f}^{\text{contact}}\left(\boldsymbol{u}, \lambda^l\right) \in \text{SPD}(n)$. This results in a sequence of linear systems of the original form in Eqs. (1): one at each nonlinear conjugate-gradient iteration. Here, $\mathbf{A}_j = \bar{\mathbf{M}}(\boldsymbol{u}_i^{(k)})$ and $\mathbf{b}_j = \nabla g_i(\boldsymbol{u}_i^{(k)})$, where the mapping between time step and nonlinear conjugate-gradient iteration is provided by $j : (i,k) \mapsto k + \sum_{\text{i}=1}^{i} K(\text{i})$. Here, $K(i)$ denotes the number of conjugate-gradient iterations needed to solve problem (62). For this problem, the total number of linear systems we consider is $p = 49$. Each of these linear systems is preconditioned using the same multigrid preconditioner $\mathbf{M}_j$ as described in Section 5.1.1; the only modification is that four levels of multigrid are used in this case due to the larger number of degrees of freedom.

5.2 Experimental setup

We implemented our proposed method in Matlab and ran experiments on a Macbook Pro with Intel 2.7 GHz i5 processor and 8 GB of RAM; the implementation performs the linear-system solves after reading in the linear systems generated by Sierra/SolidMechanics as described in Section 5.1.

For all problems, we test our framework using a full-orthogonalization method (FOM) rather than the conjugate-gradient recurrence; this amounts to replacing Step 14 in Algorithm 1 with the following [24]:

$\mathbf{p}^{(k+1)} = \mathbf{z}^{(k+1)} - \mathbf{Y}\boldsymbol{\mu}^{(k+1)}$
for $i = 1, \ldots, k$ **do**
  $\beta^{(k+1),i} = \frac{(\mathbf{r}^{(k+1)})^T \mathbf{z}^{(k+1)}}{(\mathbf{r}^{(i)})^T \mathbf{z}^{(i)}}$
  $\mathbf{p}^{(k+1)} = \mathbf{p}^{(k+1)} + \beta^{(k+1),i} \mathbf{p}^{(i)}$
end for

In exact arithmetic, this modification does not affect the result. However, in finite precision, this modification ensures that that the basis $\mathbf{V}_j$ is full rank; this is sometimes necessary to ensure nonsingular systems during stage-1 solves. Removing the effect of possible rank deficiency from the numerical experiments simplifies interpretation of the results.

Iterative-solver performance can be measured in three primary ways: the number of incurred matrix–vector products, the number of stage-3 iterations (which is equivalent to the number of preconditioner applications), and the wall time per linear-system solve. While the wall time is the most important metric in practice, we report all three metrics (in terms of their averages over all linear systems) to provide a more complete picture of performance, as the specific linear system and choice of preconditioner can have a strong effect on the relative cost of operations.

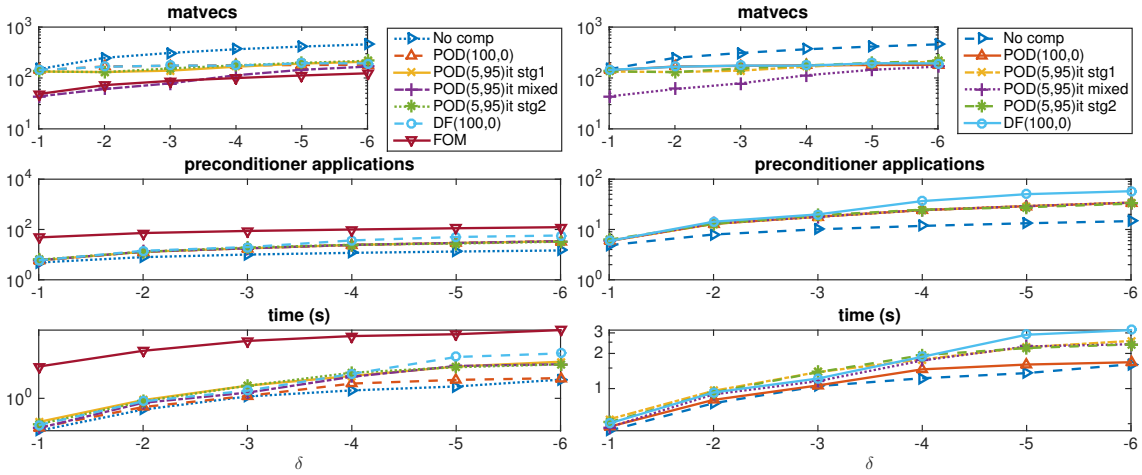
5.3 Method comparison: Pancake problem

This section compares the following methods:

1. *FOM.* This approch simply solves each linear system independently without recycling using the full orthogonalization method.
2. *No compression.* This approach does not perform truncation, and employs $\bar{y} = \infty$, $\varrho = 1$, and $\varphi = 0$ in Algorithm 3. Because it employs all Krylov vectors as the augmenting-subspace basis, it requires the fewest stage-3 iterations and, thus, the fewest number of preconditioner applications; however, the memory costs and orthogonalization costs are the largest for this method.
3. *DF(100,0).* This is the standard approach for deflation-based truncation, which places all augmenting-subspace basis vectors in the stage-1 basis. Algorithm 3 parameters are $\bar{y} = 200$, $\varrho = 1$, $\varphi = 0$, and $\mathbf{W}_{j+1} = \begin{bmatrix}[\mathbf{Y}_{j+1}]_1 & \cdots & [\mathbf{Y}_{j+1}]_{100}\end{bmatrix}$ in step 28. In step 26, the augmenting space is computed by solving Eq. (22) for $j \leftarrow j+1$ and setting $\mathbf{Y}_{j+1} \leftarrow [\mathbf{Z}_j\mathbf{g}_1 \ \cdots \ \mathbf{Z}_j\mathbf{g}_{100}]$.
4. *POD(100,0).* This approach employs POD truncation and places augmenting-subspace basis vectors in the stage-1 basis. Algorithm 3 parameters are $\bar{y} = 200$, $\varrho = 1$, $\varphi = 0$, $\mathbf{Y}_{j+1} = \mathbf{U}_{100}^{\mathbf{A}_j}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\text{RBF}}(j - \bar{j})\right)$ in step 26, and $\mathbf{W}_{j+1} = \begin{bmatrix}[\mathbf{Y}_{j+1}]_1 & \cdots & [\mathbf{Y}_{j+1}]_{100}\end{bmatrix}$ in step 28.
5. *POD(5,95)it stg1.* This approach employs POD truncation, places only the dominant POD modes in the stage-1 basis, and places all post-truncation Krylov vectors in the stage-1 basis. Algorithm 3 parameters are $\bar{y} = 200$, $\varrho = 1$, $\varphi = 1$, $\mathbf{Y}_{j+1} = \mathbf{U}_{100}^{\mathbf{A}_j}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\text{RBF}}(j - \bar{j})\right)$ in step 26, and $\mathbf{W}_{j+1} = \begin{bmatrix}[\mathbf{Y}_{j+1}]_1 & \cdots & [\mathbf{Y}_{j+1}]_5\end{bmatrix}$ in step 28. Note that $\hat{\epsilon}_j = 10^{-4}\epsilon_j$ for $\epsilon_j \geq 10^{-3}$, and $\hat{\epsilon}_j = 10^{-5}\epsilon_j$ otherwise and $\bar{\epsilon}_j = 10^{-2}\epsilon_j$ for all tolerances.

6. *POD(5,95)it mixed.* This approach employs POD truncation, places only the dominant POD modes in the stage-1 basis, and places only the dominant post-truncation Krylov vectors in the stage-1 basis. Algorithm 3 parameters are employs $\bar{y} = 200$, $\varrho = 1 \times 10^{-3}$, $\varphi = 1$, $\mathbf{Y}_{j+1} = \mathbf{U}_{100}^{\mathbf{A}_j}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(j-\bar{j})\right)$ in step 26, and $\mathbf{W}_{j+1} = \left[[\mathbf{Y}_{j+1}]_1 \ \cdots \ [\mathbf{Y}_{j+1}]_5\right]$ in step 28. Note $\hat{\epsilon}_j = 10^{-4}\epsilon_j$ for $\epsilon_j \geq 10^{-2}$, and $\hat{\epsilon}_j = 10^{-6}\epsilon_j$ otherwise and $\bar{\epsilon}_j = 10^{-2}\epsilon_j$ for all ϵ_j.

7. *POD(5,95)it stg2.* This approach employs POD truncation, places only the dominant POD modes in the stage-1 basis, and places none of the post-truncation Krylov vectors in the stage-1 basis. Algorithm 3 parameters are employs $\bar{y} = 200$, $\varrho = 0$, $\varphi = 1$, $\mathbf{Y}_{j+1} = \mathbf{U}_{100}^{\mathbf{A}_j}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(j-\bar{j})\right)$ in step 26, and $\mathbf{W}_{j+1} = \left[[\mathbf{Y}_{j+1}]_1 \ \cdots \ [\mathbf{Y}_{j+1}]_5\right]$ in step 28. Note $\hat{\epsilon}_j = 10^{-4}\epsilon_j$ for $\epsilon_j \geq 10^{-2}$, and $\hat{\epsilon}_j = 10^{-7}\epsilon_j$ otherwise. $\bar{\epsilon}_j = 10^{-2}\epsilon_j$ for $\epsilon_j \geq 10^{-3}$ and $\bar{\epsilon}_j = 10^{-3}\epsilon_j$ otherwise.



(a) All methods (b) Recycling methods only

Figure 4: Results for the pancake problem: average number of matrix–vector products, preconditioner applications, and wall time to compute solutions within tolerances $\epsilon_j = 10^{-1}$ through $\epsilon_j = 10^{-6}$.

Figure 4a reports results for all tested methods. First, we note that applying the AMG preconditioner is computationally expensive for this problem, especially relative to matrix–vector products. Therefore, there is a strong relationship between the wall-time performance and the preconditions-application performance of iterative methods in this section. Next, we note that recycling provides a significant benefit, as applying FOM without recycling is the slowest method for all tested tolerances. To more clearly distinguish the differences between recycling methods, Figure 4b reports the same results with the FOM results removed. Here, we see that the no-compression case yields the best performance as measured in both wall time and preconditioner applications; however, it performs the worst in matrix–vector products. This arises from two primary effects: (1) the preconditioner application is the dominant cost for this problem, so minimizing the number of stage-3 iterations—which will always occur by not truncating the augmenting subspace—yields the best wall time performance, and (2) the problem is small scale, so there is not a significant penalty to retaining all Krylov vectors. We also note that the POD methods (especially the POD(100,0) method) perform similarly to the no compression method; this suggests that POD truncation effectively captures the most important subspace from the set of available vectors.

Next, we note that the 'inner' iterative method described in Section 4.3.1—which is used by the three POD(5,95) methods—produces the same number of preconditioner applications as POD(100,0). This illustrates that the inner iterative method has successfully orthogonalized against the entire augmenting subspace. Figure 5 illustrates this point further by comparing three methods: the POD(5,95)it stg1 method, a variant of POD(5,95)it stg1 that employs $\varphi = 0$ in Algorithm 3, and the POD(100,0) method. While POD(5,95)it stg1 matches the preconditioning applications of POD(100,0), this comes at the cost of additional matrix–vector products for stricter tolerances; further these matrix–vector products are generally not applied as a block as in POD(100,0), which makes them more expensive on average. In addition, the cost of repeatedly multiplying vectors by the full augmenting-subspace basis $\mathbf{Y}_j$ within the inner iterative method is not counted toward matrix–vector products, even though this operation incurs

a non-negligible cost due to the density of $\mathbf{Y}_j$. As a result, POD(100,0) yields the lowest wall time; we might expect POD(5,95)it stg1 to produce a lower wall time when the dimension of the augmenting subspace is larger.

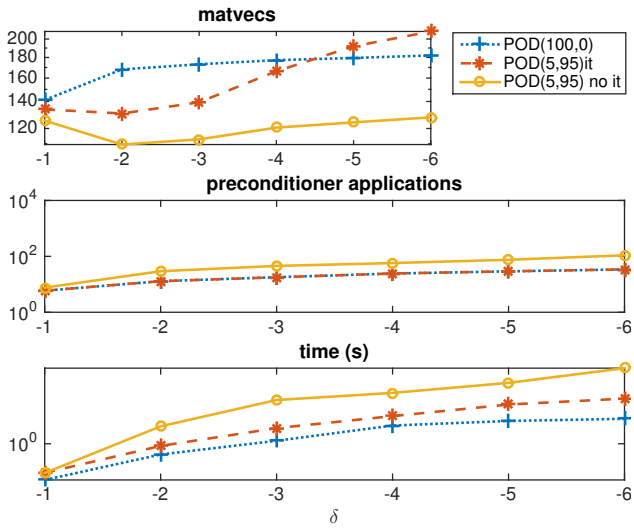


Figure 5: Comparison of iterative and non-iterative Stage 3 for the problem 1.

As shown in Theorem 4, the reduced matrix $\mathbf{Y}_j^T \mathbf{A}_j \mathbf{Y}_j$ should be well conditioned if the system matrices do not vary significantly. Figure 6 illustrates this: the condition number of the reduced system is close to one for all linear systems. This implies fast convergence of stage 2. Note that this effect comes 'for free' by employing a POD metric of $\mathbf{A}_{\bar{j}}$, as the basis is automatically $\mathbf{A}_{\bar{j}}$-orthogonal; this precludes the need for orthogonalization step 27 in Algorithm 3, as was noted in Section 4.4.1.

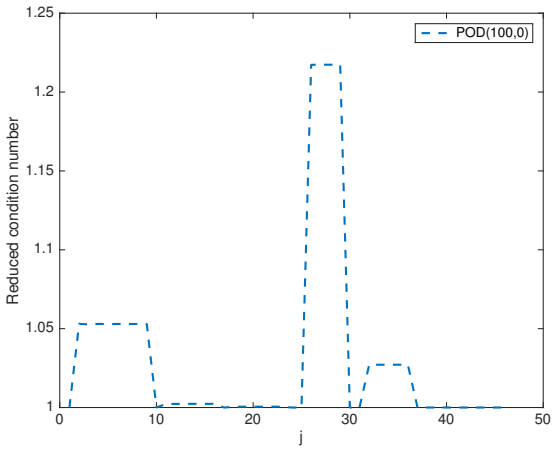


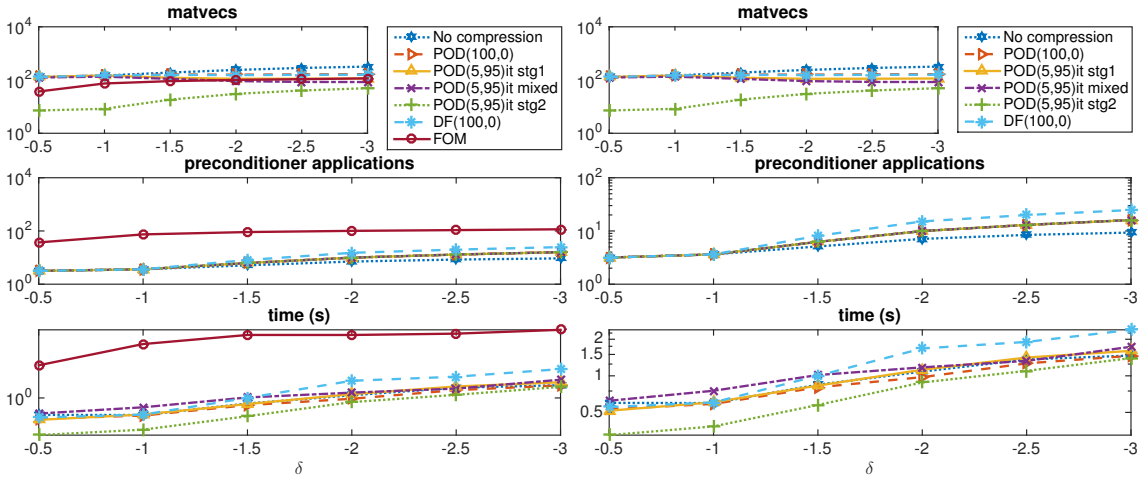
Figure 6: Condition number for reduced linear systems.

5.4 Method comparison: I-beam problem

We compare the same methods as in Section 5.3; the only modification is that all POD(5,95) methods employ parameters $\hat{\epsilon}_j = 10^{-4}\epsilon_j$, $\bar{\epsilon}_j = 10^{-2}\epsilon_j$ in Algorithm 3.

Figure 7a reports results for all tested methods. Again, we first note that the use of recycling leads to significant improvements, as the FOM method (without recycling) produces the largest wall time and requires the largest number of preconditioner applications. To more easily distinguish the relative

merits of the recycling methods, Figure 7b reports the results without FOM. This figure illustrates the need for trunction within recycling. As for the pancake problem, the 'no truncation' case minimizes the number of preconditioner applications; however, the lower matrix–vector multiplication cost and the lower overhead in the stage-3 orthogonalization steps lead to lower overall costs for several truncation methods. Figure 7b also illustrates the benefit of using the hybrid direct/iterative approach to solve over the augmenting subspace, as POD(5,95)it stg2—which employs this approach—produces the lowest wall time and number of matrix–vector products for all tested tolerances. We also note that all POD-based truncation methods outperform deflation in terms of preconditioner applications; all POD methods except for POD(5,95)it mixed also outperform deflation in terms of matrix–vector multiplications and wall time.



(a) All methods

(b) Recycling methods only

Figure 7: Results for the I-beam problem: average number of matrix–vector products, preconditioner applications, and wall time to compute solutions within tolerances $\epsilon_j = 10^{-0.5}$ through $\epsilon_j = 10^{-3}$.

Finally, Figure 8 assess the performance of the 'inner' iterative method proposed in Section 4.3.1, which aims to orthogonalize against the entire augmenting subspace within stage 3 using an iterative method. These data show that the iterative method in POD(5,95)it stg1 successfully matches the number of stage-3 iterations (i.e., preconditioner applications) as POD(100,0), yet it incurs far fewer matrix–vector products. In fact, the number of matrix–vector products is close to that realized by POD(5,95) stg1 without the inner iterative method.

5.5 POD-weights experiments

This section compares the performance of POD-based truncation methods when various POD weights are employed (see Section 3.3.2). To assess this, POD compression is performed after the solution of the first ten linear systems, and performance of the resulting truncated augmented subspace is assessed for the eleventh linear system. That is, we propose computing the truncated augmenting subspace according to

$$\mathcal{Y}_{11} = \mathcal{U}_{y_{11}}^{\mathbf{A}_{10}}(\mathbf{Z}_{10}, \boldsymbol{\gamma}) \tag{63}$$

for various choices of $\boldsymbol{\gamma}$. We compare the following approaches:

1. *Ideal weights.* This case employs $\boldsymbol{\gamma} = \boldsymbol{\eta}_{11}^{\star}$ as defined in Eq. (29). Although this weighting scheme is not practical, it illustrates the best possible choice.
2. *Previous weights.* This case employs $\boldsymbol{\gamma} = \boldsymbol{\eta}_{11}^{\text{prev}}$ as defined in Eq. (41).
3. *Radial-basis-function weights.* This case employs $\boldsymbol{\gamma} = \boldsymbol{\eta}_{11}^{\text{RBF}}(11)$ as defined in Eq. (45). This is the choice used by the previous experiments in this section.

of the POD weights proposed in Section 3.3.2.

Figure 9a illustrates that the ideal weights minimize the residual after the reduced system is solved, which implies that the ideal weights lead to a better estimate of the solution in the augmenting subspace.

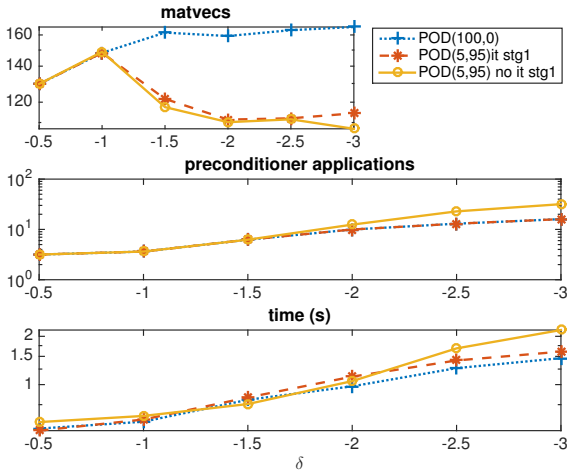
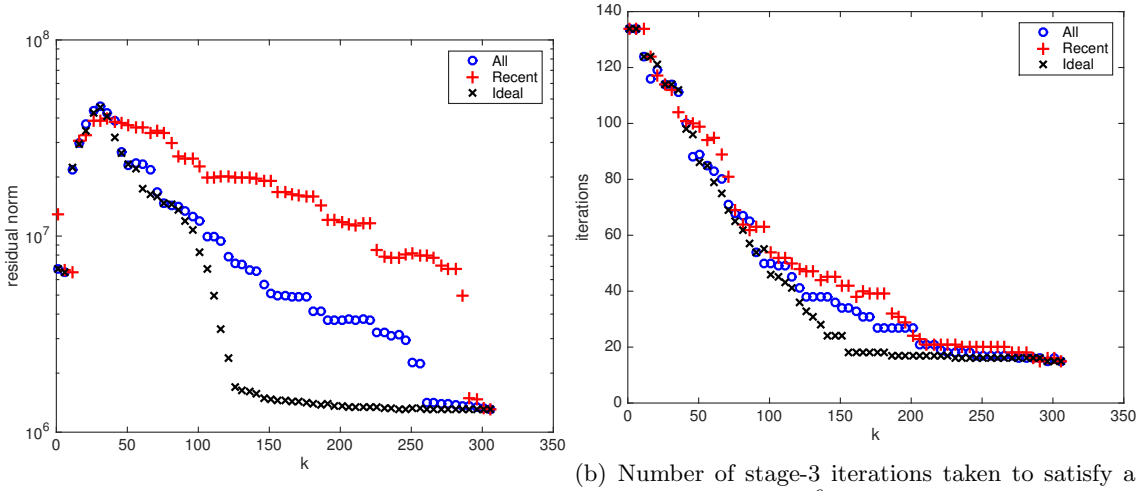


Figure 8: Results for the I-beam problem: POD-method performance in terms of average number of matrix–vector products, preconditioner applications, and wall time to compute solutions within tolerances $\epsilon_j = 10^{-0.5}$ through $\epsilon_j = 10^{-3}$.



(a) The residual norm before stage 3 as a function of the number of POD vectors k.

(b) Number of stage-3 iterations taken to satisfy a tolerance of $\epsilon_{11} = 10^{-6}$ as a function of the number of POD vectors k.

Figure 9: POD-weights experiments for the pancake problem. Results correspond to solving linear system eleven after performing POD-based compression for linear system ten using radial-basis-function weights ('All'), previous weights ('Recent'), and ideal weights ('Ideal').

Note that the radial-basis-function weights (which was employed for POD results in previous sections) yields results that are close to the ideal case. Figure 9b shows that the ideal weights also minimize the number of stage-3 iterations, while the two other methods produce a similar number of stage-3 iterations as the ideal-weights case. This suggests that radial-basis-function weights provide a good approximation of the ideal weights for both producing an accurate solution in the augmenting subspace and yielding similar stage-3 convergence.

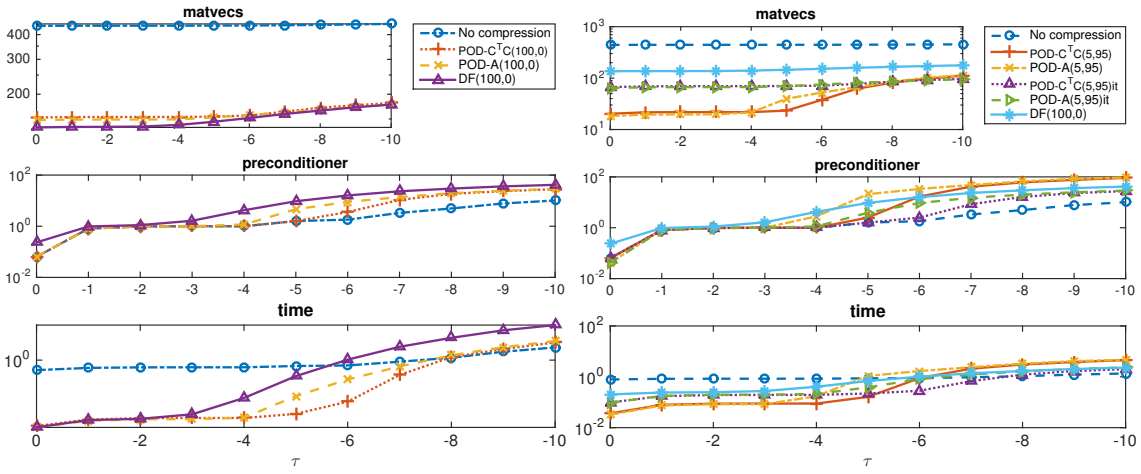
5.6 Output-oriented POD experiments

This section assesses the performance of output-oriented POD, i.e., when the metric is set to $\mathbf{\Theta} = \mathbf{C}^T\mathbf{C}$ as proposed in Section 3.3.3. For this purpose, we consider a set of $q = 100$ output quantities of interest that are random linear functionals of the solution; as such $\mathbf{C} \in [0,1]^{100\times n}$ with entries drawn from a uniform distribution in the interval $[0,1]$. We then compare the following methods:

1. *No compression.* This approach does not perform truncation, and employs $\bar{y} = \infty$, $\varrho = 1$, and $\varphi = 0$ in Algorithm 3.
2. *DF(100,0).* This is the standard approach for deflation-based truncation; Algorithm 3 parameters are $\bar{y} = 200$, $\varrho = 1$, $\varphi = 0$, and $\mathbf{W}_{j+1} = \left[[\mathbf{Y}_{j+1}]_1 \ \cdots \ [\mathbf{Y}_{j+1}]_{100}\right]$ in step 28. In step 26, the augmenting space is computed by solving Eq. (22) for $j \leftarrow j+1$ and setting $\mathbf{Y}_{j+1} \leftarrow [\mathbf{Z}_j\mathbf{g}_1 \ \cdots \ \mathbf{Z}_j\mathbf{g}_{100}]$.
3. *POD-A(100,0).* This approach employs POD truncation and places augmenting-subspace basis vectors in the stage-1 basis. Algorithm 3 parameters are $\bar{y} = 200$, $\varrho = 1$, $\varphi = 0$, $\mathbf{Y}_{j+1} = \mathbf{U}_{100}^{\mathbf{A}_j}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(j-\bar{j})\right)$ in step 26, and $\mathbf{W}_{j+1} = \left[[\mathbf{Y}_{j+1}]_1 \ \cdots \ [\mathbf{Y}_{j+1}]_{100}\right]$ in step 28.
4. *POD(5,95).* This approach employs POD truncation, places only the dominant POD modes in the stage-1 basis, and places all post-truncation Krylov vectors in the stage-1 basis. Algorithm 3 parameters are $\bar{y} = 200$, $\varrho = 1$, $\varphi = 0$, $\mathbf{Y}_{j+1} = \mathbf{U}_{100}^{\mathbf{A}_j}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(j-\bar{j})\right)$ in step 26, and $\mathbf{W}_{j+1} = \left[[\mathbf{Y}_{j+1}]_1 \ \cdots \ [\mathbf{Y}_{j+1}]_5\right]$ in step 28.
5. *POD(5,95)it.* This approach employs POD truncation, places only the dominant POD modes in the stage-1 basis, and places all post-truncation Krylov vectors in the stage-1 basis. Algorithm 3 parameters are $\bar{y} = 200$, $\varrho = 1$, $\varphi = 1$, $\mathbf{Y}_{j+1} = \mathbf{U}_{100}^{\mathbf{A}_j}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(j-\bar{j})\right)$ in step 26, and $\mathbf{W}_{j+1} = \left[[\mathbf{Y}_{j+1}]_1 \ \cdots \ [\mathbf{Y}_{j+1}]_5\right]$ in step 28. Note that $\bar{\epsilon}_j = 10^{-2}\epsilon_j$.
6. *POD-C^TC(100,0).* This approach employs POD truncation and places augmenting-subspace basis vectors in the stage-1 basis. Algorithm 3 parameters are $\bar{y} = 200$, $\varrho = 1$, $\varphi = 0$, $\mathbf{Y}_{j+1} = \mathbf{U}_{100}^{\mathbf{C}^T\mathbf{C}}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(j-\bar{j})\right)$ in step 26, and $\mathbf{W}_{j+1} = \left[[\mathbf{Y}_{j+1}]_1 \ \cdots \ [\mathbf{Y}_{j+1}]_{100}\right]$ in step 28.
7. *POD-C^TC(5,95).* This approach employs POD truncation, places only the dominant POD modes in the stage-1 basis, and places all post-truncation Krylov vectors in the stage-1 basis. Algorithm 3 parameters are $\bar{y} = 200$, $\varrho = 1$, $\varphi = 0$, $\mathbf{Y}_{j+1} = \mathbf{U}_{100}^{\mathbf{C}^T\mathbf{C}}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(j-\bar{j})\right)$ in step 26, and $\mathbf{W}_{j+1} = \left[[\mathbf{Y}_{j+1}]_1 \ \cdots \ [\mathbf{Y}_{j+1}]_5\right]$ in step 28.
8. *POD(5,95)it.* This approach employs POD truncation, places only the dominant POD modes in the stage-1 basis, and places all post-truncation Krylov vectors in the stage-1 basis. Algorithm 3 parameters are $\bar{y} = 200$, $\varrho = 1$, $\varphi = 1$, $\mathbf{Y}_{j+1} = \mathbf{U}_{100}^{\mathbf{C}^T\mathbf{C}}\left(\mathbf{Z}_j, \boldsymbol{\eta}_j^{\mathrm{RBF}}(j-\bar{j})\right)$ in step 26, and $\mathbf{W}_{j+1} = \left[[\mathbf{Y}_{j+1}]_1 \ \cdots \ [\mathbf{Y}_{j+1}]_5\right]$ in step 28. Note that $\bar{\epsilon}_j = 10^{-2}\epsilon_j$.

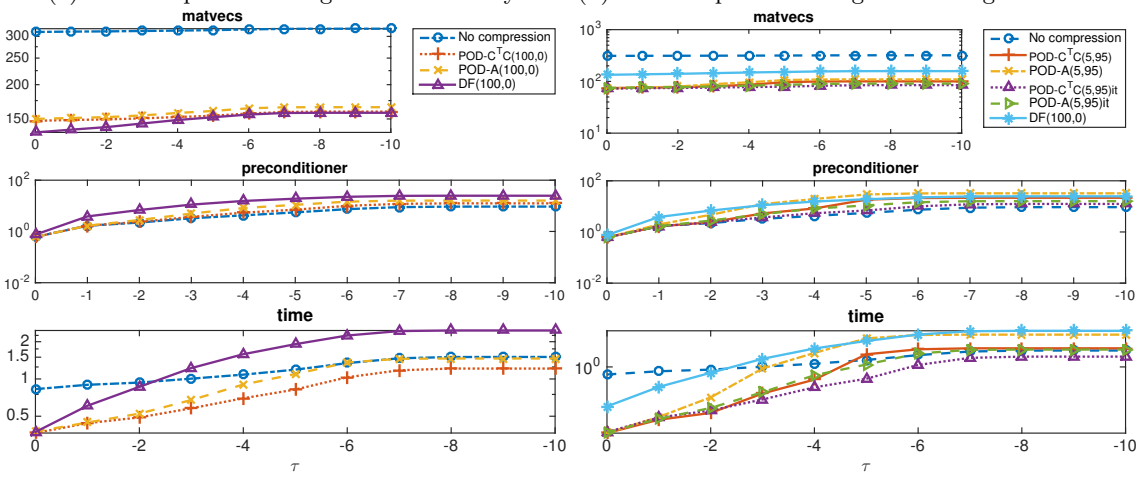
To assess the performance of output-oriented POD truncation—which aims to accurately represent the output quantity of interest—we monitor the error of the solution in the output-oriented norm $||\mathbf{x}_j^\star - \mathbf{x}_j^{(k)}||_{\mathbf{C}^T\mathbf{C}}$. For the pancake problem, we employ $\epsilon_j = 10^{-6}$, while we use $\epsilon_j = 10^{-3}$ for the I-beam problem. During the execution of the stage-3 algorithm, we track the output-oriented error and report the average number of matrix–vector products, preconditioner applications, and wall time for the error to satisfy $||\mathbf{x}_j^\star - \mathbf{x}_j^{(k)}||_{\mathbf{C}^T\mathbf{C}} < \tau$ for a variety of output-oriented tolerance τ.

Figures 10a–10d compare the performances of the assessed methods as a function of the output-oriented tolerance τ. First, note that Figures 10a–10b show that POD-based truncation methods produce the lowest wall times for *inexact output-oriented tolerances* for the pancake problem. These plots also illustrate the benefit of output-oriented metric over the $\mathbf{A}_j$-metric: Figure 10a shows the superior performance of the former for modest goal-oriented tolerances, i.e., those between $\tau = 10^{-4}$ and 10^{-7}. Figure 10b shows that the POD(5,95) methods without the inner iterative method described in Section 4.3.1 are faster for inexact goal-oriented tolerances; this occurs because very few stage-3 iterations



(a) Pancake problem: stage-1 methods only.

(b) Pancake problem: stage-1 and stage-2 methods.



(c) I-beam problem: stage-1 methods only.

(d) I-beam problem: stage-1 and stage-2 methods.

Figure 10: Output-oriented truncation experiments: average number of matrix–vector products, preconditioner applications, and wall time to compute solutions as a function of the goal-oriented tolerance τ.

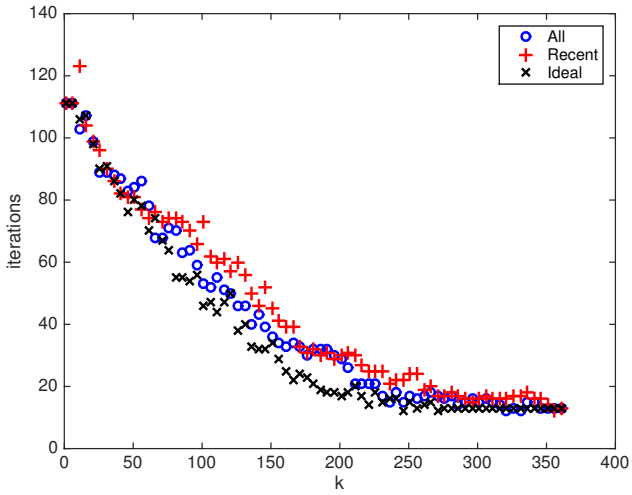


Figure 11: Number of stage-3 iterations taken to satisfy an output-oriented tolerance of $\tau = 10^{-6}$ for the pancake problem as a function of the augmenting-subspace dimension k for radial-basis-function weights ('All'), previous weights ('Recent'), and ideal weights ('Ideal').

are required for convergence in the output-oriented norm to these tolerances. In contrast, for stricter tolerances $\tau \leq 10^{-6}$, the inner iterative modification (or no compression) yields superior performance.

Figures 10c–10d present results for the I-beam problem. These results illustrate the advantage to using the output-oriented metric for most values of the goal-oriented tolerance τ, as POD-C^TC(100,0) produces the lowest wall time in Figure 10c and the POD-C^TC(5,95)it method produces the lowest wall time for $\tau \leq 10^{-3}$.

Finally, we repeat the POD-weights experiments discussed in Section 5.5 for the output-oriented POD-based truncation method applied to the pancake problem. The experiments measure the number of stage-3 iterations required to converge to a output-oriented tolerance of $\tau = 10^{-6}$ as a function of the dimension of the augmenting subspace. We employ a stage-3 tolerance of $\epsilon_j = 10^{-8}$ to accrue the search directions for the first 10 linear systems and subsequently perform truncation using the output-oriented POD metric with different POD weights. The resulting stage-3 iterations are for the eleventh linear system. As we observed for the $\mathbf{A}_j$-metric, Figure 11 shows that the ideal weights yield the best performance, as they minimizing the number of stage-3 iterations required for convergence in the goal-oriented norm, while the other weighting schemes closely follow the ideal weights.

6 Conclusions

This work has proposed a novel strategy for Krylov-subpace recycling inspired by goal-oriented proper orthogonal decomposition (POD). We performed analyses that exposes the close connection between model reduction and Krylov-subspace recycling, proposed specific goal-oriented POD ingredients for truncating previous Krylov vectors, and developed a new 'three-stage' algorithm that employs a hybrid direct/iterative approach for efficiently solving over the augmenting and Krylov subspaces. Results on several solid-mechanics problems highlighted the benefits of the new contributions, especially for capturing specific output quantities of interest to modest tolerances.

Acknowledgments

We thank Paul Tsuji for his contributions to obtaining the linear systems used in the numerical experiments, as well as Kendall Pierson for his assistance in using Adagio. We also thank Howard Elman for both insightful conversations and his support in establishing this collaboration. K. Carlberg acknowledges an appointment to the Sandia National Laboratories Truman Fellowship in National Security Science and Engineering. The Truman Fellowship is sponsored by Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a

wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. The content of this publication does not necessarily reflect the position or policy of any of these institutions, and no official endorsement should be inferred.

A POD computation

This section describes two techniques for computing a POD basis using given snapshots, weights, and a pseudometric (see Ref. [4] for additional details). Algorithm 4 describes the first technique, which is based on the eigenvalue decomposition and is equivalent to the well-known "method of snapshots" [26]. Algorithm 5 reports the second case, which is based on the singular value decomposition (SVD) and is more appropriate when the symmetric factorization $\mathbf{\Theta} = (\mathbf{\Theta}^{1/2})^T\mathbf{\Theta}^{1/2}$ is available, where $\mathbf{\Theta}^{1/2}$ need not be upper triangular; this approach leads to a more well-conditioned linear system. Note that Algorithms 4 and 5 produce equivalent POD bases (in exact arithmetic) and differ only in their first two steps.

Algorithm 4 `pod_evd`

Input: snapshot matrix $\mathbf{W} \in \mathbb{R}^{n\times s}$, weights $\boldsymbol{\gamma} \in \mathbb{R}^s$, pseudometric $\mathbf{\Theta} \in \mathrm{SPSD}(n)$, and energy criterion $\varepsilon \in [0,1]$

Output: POD basis $\mathbf{U}_y^{\mathbf{\Theta}}(\mathbf{W}, (\gamma_1, \ldots, \gamma_s))$

1: $\bar{\mathbf{\Theta}} = \mathrm{diag}(\gamma_1, \ldots, \gamma_s)\,\mathbf{W}^T\mathbf{\Theta}\mathbf{W}\,\mathrm{diag}(\gamma_1, \ldots, \gamma_s)$
2: Compute symmetric eigenvalue decomposition $\bar{\mathbf{\Theta}} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T$
3: Choose dimension of truncated basis $y = \min_{i\in A} i$ with $A := \{i \in \mathbb{N}(\mathrm{rank}(\bar{\mathbf{\Theta}})) \mid \sum_{k=1}^{i}\sigma_k^2 / \sum_{\ell=1}^{s}\sigma_\ell^2 \geq \varepsilon\}$
4: $\mathbf{U}_y^{\mathbf{\Theta}}(\mathbf{W}, (\gamma_1, \ldots, \gamma_s)) = \mathbf{W}\left[\frac{1}{\sigma_1}\mathbf{v}_1 \;\; \ldots \;\; \frac{1}{\sigma_y}\mathbf{v}_y\right]$, where $\mathbf{\Sigma} := \mathrm{diag}(\sigma_1, \ldots, \sigma_{n_w})$

Algorithm 5 `pod_svd`

Input: snapshot matrix $\mathbf{W} \in \mathbb{R}^{n\times s}$, weights $\boldsymbol{\gamma} \in \mathbb{R}^s$, pseudometric factor $\mathbf{\Theta}^{1/2}$ such that $\mathbf{\Theta} = (\mathbf{\Theta}^{1/2})^T\mathbf{\Theta}^{1/2} \in \mathrm{SPSD}(n)$, and energy criterion $\varepsilon \in [0,1]$

Output: POD basis $\mathbf{U}_y^{\mathbf{\Theta}}(\mathbf{W}, (\gamma_1, \ldots, \gamma_s))$

1: $\bar{\mathbf{W}} = \mathbf{\Theta}^{1/2}\mathbf{W}$
2: Compute thin singular value decomposition $\bar{\mathbf{W}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$
3: Choose dimension of truncated basis
4: Choose dimension of truncated basis $y = \min_{i\in A} i$ with $A := \{i \in \mathbb{N}(\mathrm{rank}(\bar{\mathbf{W}})) \mid \sum_{k=1}^{i}\sigma_k^2 / \sum_{\ell=1}^{s}\sigma_\ell^2 \geq \varepsilon\}$
5: $\mathbf{U}_y^{\mathbf{\Theta}}(\mathbf{W}, (\gamma_1, \ldots, \gamma_s)) = \mathbf{W}\left[\frac{1}{\sigma_1}\mathbf{v}_1 \;\; \ldots \;\; \frac{1}{\sigma_y}\mathbf{v}_y\right]$, where $\mathbf{\Sigma} := \mathrm{diag}(\sigma_1, \ldots, \sigma_{n_w})$

References

[1] J. Bishop, J. Emery, R. Field, C. Weinberger, and D. Littlewood. Direct numerical simulations in solid mechanics for understanding the macroscale effects of microscale material variability. *Computer Methods in Applied Mechanics and Engineering*, 287:262–289, 2015.

[2] K. Carlberg and C. Farhat. A compact proper orthogonal decomposition basis for optimization-oriented reduced-order models. *AIAA Paper 2008-5964, 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, Canada*, September 10–12, 2008.

[3] K. Carlberg and C. Farhat. An adaptive POD-Krylov reduced-order model for structural optimization. *8th World Congress on Structural and Multidisciplinary Optimization, Lisbon, Portugal*, June 1–5 2009.

[4] K. Carlberg and C. Farhat. A low-cost, goal-oriented 'compact proper orthogonal decomposition' basis for model reduction of static systems. *International Journal for Numerical Methods in Engineering*, 86(3):381–402, April 2011.

[5] A. Chapman and Y. Saad. Deflated and augmented Krylov subspace techniques. *Numerical Linear Algebra with Applications*, 4(1):43–66, 1997.

[6] E. De Sturler. Truncation strategies for optimal Krylov subspace methods. *SIAM Journal of Numerical Analysis*, 36(3):864–889, 1999.

[7] H Carter Edwards, Alan B Williams, Gregory D Sjaardema, David G Baur, and William K Cochran. Sierra toolkit computational mesh conceptual model. *Sandia National Laboratories SAND Series, SAND*, 1192:2010, 2010.

[8] S.C. Eisenstat and H.F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM J. Sci. Comput*, 1996.

[9] J. Erhel and F. Guyomarc'h. An augmented conjugate gradient method for solving consecutive symmetric positive definite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1279–1299, 2000.

[10] C. Farhat. Optimizing substructuring methods for repeated right hand sides, scalable parallel coarse solvers, and global/local analysis. *Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering*, pages 141–160, 1995.

[11] C. Farhat, L. Crivelli, and F. X. Roux. Extending substructure based iterative solvers to multiple load and repeated analyses. *Computer Methods in Applied Mechanics and Engineering*, 117:195–209, July 1994.

[12] C. Farhat, K. Pierson, and M. Lesoinne. The second generation FETI methods and their application to the parallel solution of large-scale linear and geometrically non-linear structural analysis problems. *Computer Methods in Applied Mechanics and Engineering*, 184(2–4):333–374, April 2000.

[13] J. A. Mitchell, A. S. Gullerud, W. M. Scherzinger, R. Koteras, and V. L. Porter. Adagio: non-linear quasi-static structural response using the SIERRA framework. 2001.

[14] R.B. Morgan. GMRES with deflated restarting. *SIAM Journal on Scientific Computing*, 24(1):20–37, 2003.

[15] Ronald B Morgan. Computing interior eigenvalues of large matrices. *Linear Algebra and its Applications*, 154:289–309, 1991.

[16] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra and its Applications*, 29:293–322, 1980.

[17] M.L. Parks, E. De Sturler, G. Mackey, D.D. Johnson, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5):1651–1674, 2007.

[18] C. Rey. *Developpement d'algorithmes paralleles de resolution en calcul non-lineaire de structures heterogenes: application au cas d'une butee acier-elastomere*. PhD thesis, Laboratoire de Modelisation et Mecanique des Structures, University of Paris VI, December 1994.

[19] C. Rey and F. Risler. A Rayleigh–Ritz preconditioner for the iterative solution to large scale nonlinear problems. *Numerical Algorithms*, 17(3):279–311, 1998.

[20] F. Risler and C. Rey. Iterative accelerating algorithms with Krylov subspaces for the solution to large-scale nonlinear problems. *Numerical Algorithms*, 23(1):1–30, 2000.

[21] F. X. Roux. *Parallel implementation of a domain decomposition method for non-linear elasticity*, pages 161–175. SIAM, 1995.

[22] Y. Saad. On the Lanczos method for solving symmetric linear systems with several right-hand sides. *Mathematics of Computation*, pages 651–662, 1987.

[23] Y. Saad. Analysis of augmented Krylov subspace methods. *SIAM Journal on Matrix Analysis and Applications*, 18(2):435–449, 1997.

[24] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[25] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h. A deflated version of the conjugate gradient algorithm. *SIAM Journal on Scientific Computing*, 21(5):1909–1926, 2000.

[26] L. Sirovich. Turbulence and the dynamics of coherent structures. I: Coherent structures. *Quarterly of Applied Mathematics*, 45:561–571, October 1987.